NORGES TEKNISK-NATURVITENSKAPELIGE UNIVERSITET
FAKULTET FOR FYSIKK, INFORMATIKK OG MATEMATIKK

HOVEDOPPGAVE

Kandidatens navn:	Geir Sørensen
Fag:	Datateknikk
Oppgavens tittel (norsk):	Tjener for digitale videoarkiv
Oppgavens tittel (engelsk):	Server for digital video archives
Oppgavens tekst:	

Institutt for datateknikk og informasjonsvitenskap har utviklet videotjeneren Elvira. Denne leverer digital MPEG- og JPEG-komprimert video over et ATM-nettverk. Videoen i videotjeneren er i dag lagret på harddisker koblet til en klynge av arbeidsstasjoner. For å bygge opp store videodatabaser, som f.eks. digitale fjernsynsarkiv, er det nødvendig å kunne lagre svært store mengder digital video, gjerne flere terabyte. Å lagre all denne videoen på harddisk er ikke økonomisk mulig i mange sammenhenger. Rimeligere lagringsmedier som f.eks. digitale bånd og optisk lager må derfor vurderes.

Oppgaven går ut på å studere ulike måter å lagre store mengder digital video og å vurdere egenskapene til de forskjellige løsningene. Spesielt interessant er det å vurdere og teste hvor godt egnet databånd er for lagring av digital video. Viktige problemstillinger er blant annet organiseringen av videolagringen for å få rask tilgang til videoen, og hvordan en best utnytter den tilgjengelige harddiskmengden som et «hurtigbuffer» for et båndbasert videoarkiv. Basert på denne studien skal det utarbeides en modell for hvordan videotjeneren Elvira kan utvides til å ta i bruk lagring på databånd. Modellen bør fokusere på god utnyttelse av tilgjengelige ressurser for å kunne betjene flest mulig brukerforespørsler samtidig som responstiden for den enkelte forespørsel holdes lav. En del av denne modellen bør implementeres og evalueres gjennom praktiske eksperimenter.

Oppgaven gitt:	1996-09-30
Besvarelsen leveres innen:	1997-03-21
Besvarelsen levert:	1997-03-21
Utført ved:	Institutt for datateknikk og informasjonsvitenskap
Veileder:	Stipendiat Olav Sandstå

Trondheim, 1997-03-21

Amanuensis Roger Midtstraum
Faglærer

Sammendrag

Denne hovedoppgaven er et resultat av et prosjekt utført for Gruppe for databaseteknikk ved Institutt for datateknikk og informasjonsvitenskap på Norges teknisk-naturvitenskapelige universitet. Tema for oppgaven er tjener for digitale videoarkiv.

Store videoarkiver, f.eks. fjernsynsarkiver, har gjerne flere terabyte med videoinformasjon lagret. Organisasjonene som eier disse arkivene ønsker å gjøre dem tilgjengelige for flere brukere ved hjelp av et datanettverk og en videotjener. Lagringsløsninger utelukkende basert på harddisk blir imidlertid altfor kostnadskrevende. En mulig løsning på dette problemet kan være at videotjeneren lagrer dataene i et digitalt båndarkiv, hvor kostnadene for lagring vil være langt mindre, og bruker harddisker kun for å buffre opp video som skal benyttes aktivt.

Hovedoppgaven er tredelt. Det skulle lages en modell av en tjener for et digitalt videoarkiv, hvor deler av denne modellen skulle konstrueres og implementeres og til slutt evalueres.

Modellen som er utviklet tar først for seg en enkelt båndstasjon med ett bånd. Denne enkle modellen utvidet med flere bånd og flere stasjoner. Disse modellene ble så utvidet til å gjelde et helt system for leveranse av video fra et digitalt båndarkiv. Der ble det diskutert hvordan de andre delene av systemet vil påvirk den totale ytelsen. Det ble introdusert kømodeller for systemet. Til slutt ble omgivelsenes påvirkning av en videoarkivtjener diskutert.

Konstruksjon og implementasjon av deler av en eksperimentell videoarkivtjener har blitt utført. Konstruksjonen består av fire hovedmoduler: (1) tertiær videopumpe, (2) bufferhåndterer, (3) ressursfordeler og (4) metadatabaser. En tertiær videopumpe har ansvar for alle dataoverføringer til og fra bånd. En bufferhåndterer kontrollerer plassforbruket på harddisk. Ressursfordeleren tar seg brukernes forespørsler om innhenting av videofilm fra arkiv til harddisker. Metadatabasene inneholder informasjon om hva som er lagret på bånd, og blir flittig brukt av de tre andre modulene. Det implementert prototyper av metadatabasene og den tertiære videopumpen.

I siste del av oppgaven er det foretatt eksperimenter med den implementerte prototypen. Det ble blant annet utført målinger av aksesstid med den tertiære videopumpen, der båndstasjonen startet å søke fra begynnelsen av båndet til forskjellige videofilmer ute på båndet. Målinger mot de samme blokkaddressene ble gjort flere ganger. Et svært interessant resultat fra disse målingene var at aksesstiden mot samme blokkaddresse varierte svært lite fra måling til måling. Dette er et resultat som kan utnyttes ved f.eks. ressursfordeling i en mer fullstendig implementasjon av en videoarkivtjener, fordi man får gode estimater på hvor lang tid en søkeoperasjon vil ta for en film som befinnner seg et vilkårlig sted på båndet.

Forord

Denne rapporten er resultatet av en hovedoppgave utført av Geir Sørensen ved Institutt for datateknikk og informasjonsvitenskap (IDI) på Norges teknisk-naturvitenskapelige universitet.

Arbeidet med båndstasjonen har blitt gjort enklere gjennom et godt samarbeid med Rune Sætre, fjerdeårsstudent ved Institutt for teknisk kybernetikk, som har jobbet for IDI ved siden av studiene. Jeg vil takke ham for gode lavnivårutiner som gjorde at jeg kunne konsentrere meg om selve håndteringen av dataene og ikke lagringen av dem.

Samarbeidet med min veileder, stipendiat Olav Sandstå, har vært godt. Vi har hatt ukentlige møter under hele oppgaveperioden der mange av ideene i denne rapporten har hatt sitt utspring, og enda flere har blitt forkastet. I tillegg har Olav vært flink til å raskt skaffe de maskinvareressursene som har blitt benyttet i arbeidet.

Jeg vil også takke faglærer Roger Midtstraum for mange nyttige innspill og kommentarer til arbeidet. Olav, Roger og Helge G. Solheim, har dessuten tatt seg bryet med å korrekturlese denne rapporten, og jeg skylder dem en ekstra takk for dette.

Til slutt vil jeg takke NRK, og spesielt Bjørn Klock, for et svært informativt møte i Oslo i desember 1996.

Geir Sørensen

Trondheim, Norge
Mars 1997

Innhold

Figurer

Tabeller

Kapittel 1

Innledning

Denne rapporten er et resultat av en hovedoppgave med formål å utvikle et system for lagring av digital video på databånd for en eksisterende videotjener. Arbeidet er utført ved Institutt for datateknikk og informasjonsvitenskap (IDI, tidligere IDT) ved Norges teknisk-naturvitenskapelige universitet (NTNU).

Hvorfor bruke databånd i en videotjener? Det finnes i dag brukere med et behov for forholdsvis rask tilgang til store mengder videomateriale via et datanettverk i f.eks. store mediebedrifter som Norsk Rikskringkasting (NRK). Dette vil kunne gjøre de ansatte i stand til å lage bedre produkter på en enklere og raskere måte. I dag foregår gjennomsyn av video for en produksjon ved at videobånd blir transportert fysisk, og mye tid blir kastet bort på å se igjennom materiale som ikke kan brukes. Denne hovedoppgaven er knyttet til en eksisterende videotjener ved IDI (Elvira) og en videotjener som er under utbygging (Elvira2). Målet er å studere og vurdere metoder for hvordan denne kan utvides til å benytte bånd for lagring av store mengder digital video og å konstruere og implementere en løsning som deretter skal testes og evalueres.

I dag blir digitale databånd stort sett brukt til følgende formål: (1) sikkerhetskopiering av store mengder data, (2) lagringsmedium for dataintensive målinger, f.eks. i forbindelse med oljeleting, medisinske undersøkelser og romferder, (3) permanent arkivering for senere generasjoner og (4) distribusjon av programvare og data. Videotjenere blir stort sett brukt til film-på-forespørsel (VOD, Video-On-Demand), der man benytter et stort antall harddisker som hovedlagringsmedium. En videotjener for et *tilgjengelig videoarkiv* forsøker å benytte det beste fra to verdener; billig masselagring og høy tilgjengelighet. Dette vil diskuteres ytterligere i følgende kapitler.

Målene med oppgaven er som følger:

1. Lage en modell for en arkivtjener som lagrer video på digitale databånd.
2. Konstruere og implementere modulen som gjør Elvira i stand til å benytte databånd.
3. Eksperimentere med båndlagringsmodulen for å evaluere modell og implementasjon.

1.1 Systemterminologi

Elvira Storage Manager (ESM) er navnet på det settet av moduler, programmer og prosesser som skal utvide lagerhierarkiet til Elvira Video Server slik at data kan lagres på databånd istedenfor harddisker. ESM blir altså et resultat av arbeidet med denne hovedoppgaven.

Elvira Catalog Manager (ECM) er en katalog for lagring av informasjon som beskriver den fysiske strukturen på innholdet i videotjeneren, det vil si at den er en database for metainformasjon. Denne ble utviklet av Sørensen og Sandstå (1996).

Elvira Video Server er en videotjener utviklet ved IDT. Hovedfundamentet er en hovedoppgave skrevet av Langørgen (1994). Denne er videreutviklet av Sandstå et al. (1996). Det foregår nå arbeid med en helt omskrevet versjon kalt Elvira2.

VideoSTAR er et eksperimentelt databaserammeverk for behandling av videoinformasjon. VideoSTAR ble utviklet av Rune Hjelsvold i forbindelse med en doktoravhandling på IDT (Hjelsvold 1995). Blant annet ble det laget en algebra for forespørsler mot videodatabaser. Originalt brukte man UNIX filsystem for å lagre digitalt videomateriale, så Elvira ble laget for å gjøre det mulig å bruke VideoSTAR over nettverk.

Når det er snakk om Elvira Video Server med ESM integrert, benyttes ofte forkortelsen Elvira/ESM.

1.2 Databegreper

Multimediadata kan f.eks. være digitaliserte audio eller videosignaler, men også stillbilder, grafikk og tekst. Dette ordet vil ofte bare forkortes til *mediadata*.

Kontinuerlige multimediadata er data som må spilles av i en bestemt rate over tid for å få noen mening. Eksempler på kontinuerlige mediadata er:

- lyd – varierende amplitude over tid utgjør signalet som mennesker oppfatter som audioinformasjon (øret oppfatter vanligvis frekvenser mellom 20 Hz og 20 kHz).
- video – en strøm av stillbilder som byttes ut over tid med i en bestemt rate (vanligvis mellom 25 og 30 ganger i sekundet).

I dag er det stort sett audio og video som går under betegnelsen kontinuerlige mediadata. I fremtiden kan det også være andre former for data som blir lagret som kontinuerlige strømmer, f.eks. motoriske data som skal brukes til å påvirke brukerne når de ser på en tredimensjonal film på en avansert kino.

Diskrete multimediadata er tekst, bilder og store binærobjekter[1]. Slike data er det støtte for å lagre i eksisterende databaser, som f.eks. Postgres (Fournier 1997).

I denne rapporten vil «multimediadata» innebære «kontinuerlige multimediadata», siden det er denne typen data som vil fokuseres i denne hovedoppgaven.

1.3 Bakgrunn

I fremtiden vil man kanskje se på det som skjedde på 90-tallet som «datanettverks-revolusjonen», på samme måte som vi i dag ser på den industrielle revolusjon på slutten av 1700-tallet. Et enormt antall mennesker har fått sin hverdag endret av denne «revolusjonen».

Fra å være et forholdsvis lite nettverk forbeholdt akademiske institusjoner, har Internettet vokst eksplosivt på 90-tallet, og i den rike delen av verden begynner en stor del av befolkningen å ha tilgang til nettverk. Den store økningen i brukere har ført til en rivende utvikling innen nettverksteknologi. På et nettverk som egentlig var beregnet for overføringer av korte tekstbeskjeder og filoverføringer av beskjeden størrelse, transporterer man nå store mengder multimediadata.

På bakgrunn av den utrolige utviklingen av nettverksteknologi, har flere sett muligheter for å utvikle maskin- og programvare for leveranse av video på forespørsel (Video On Demand, VOD) og interaktivt fjernsyn (Laursen et al. 1994, Bolosky et al. 1996, Federighi og Rowe 1994). Dette er stort sett systemer som, selv om datamengdene er store, likevel kan lagres på harddisker. Undersøkelser

[1] ofte kalt Binary Large Objects (BLOBS)

har vist at harddisker er mest kostnadseffektivt med hensyn på antall parallelle videostrømmer en videotjener kan håndtere (Chervenak et al. 1995).

Samtidig har det foregått forskning på hvordan man skal behandle video i databasesammenheng (Hjelsvold 1995, Rowe et al. 1994). Noen konklusjoner av denne forskningen er:

- Materiale som tidligere kun ble sett på som «en stor klump» får nå en formell strukturbeskrivelse på samme måte som f.eks. heltall og strenger har i eksisterende relasjonsdatabaser. Dette muliggjør blant annet deling av video og metadata, slik at man unngår overflødig lagring.
- Bibliografidata kan knyttes mot video. Dette gir informasjon om produsent, tid, medarbeidere, rettigheter, sjanger og annet.
- Introduksjon av innholdsbeskrivende data, f.eks. nøkkelrammer som representerer viktige bilder i videostrømmen, angivelser av når viktige aktører dukker opp i videoen og når de forsvinner igjen. Dette åpner for enklere søking og gjennomsyn for store mengder digital video.

Mange organisasjoner, institusjoner og bedrifter sitter med store mengder materiale lagret over flere titalls år (Noreld 1996). Disse dataene er i dag vanskelig tilgjengelige, fordi materialet må flyttes fysisk, med alle de ulemper det innebærer. Ved å bruke datanettverk med høy kapasitet kan denne situasjonen endres. En ren VOD-tjener vil være uaktuelt, siden en slik tjener er spesialisert for mange kunder mot en relativt begrenset antall mengde filmer. Det ville blitt unødvendig dyrt å lagre alt materialet på harddisker, siden bare en liten del vil bli aksessert ofte. Et digitalt videoarkiv som kan betjene kunder via et slikt nettverk vil være et aktivum for en slik organisasjon. Slike organisasjoner er allerede på god vei til å digitalisere videomaterialet sitt. Digital lagring gjør at det såkalte generasjonstapet man får ved kopiering fra analoge til analoge media forsvinner. Digitalt videomateriale kan kopieres uten tap fra kopi til kopi.

Et digitalt videoarkiv gir en unik mulighet til å organisere materialet i en databaselignende struktur. Dette gir følgende fordeler:

- Man kan utføre avanserte søk og spare tid fordi man lettere finner det man leter etter, uten å måtte se gjennom materiale.
- Det blir mulig å navigere rundt i arkivet fra en hvilken som helst datamaskin koplet til høykapasitetsnettverket videotjeneren står på.
- Med full oversikt over materialet man besitter, er det lettere å kontrollere at det er lagret sikkert.
- Det blir mulig å produsere nye videodokumenter ved hjelp av materiale som allerede er lagret i arkivet.

Elvira er en videotjener utviklet på IDT (nå IDI), som baserer seg på lagring av video på harddisk (Sandstå et al. 1996, Langørgen 1994). Hovedbestanddelene i Elvira er videopumper som kan overfører data fra harddisk til klienter via nettverk i sanntid og en kontrollprosess som mottar henvendelser fra klienter som ønsker å motta video og setter opp videopumper for å betjene forespørslene.

Elvira er for tiden et system som er i sterk endring. IDI samarbeider med Telenor forskning og utvikling om konstruksjon og implementasjon av en ny versjon, kalt Elvira2. Denne blir bygd fra bunnen av, og skiller seg fra den første versjonen på følgende områder:

- Konstruksjonen er enklere → Mer effektiv leveranse av video og bedre utnyttelse av ressurser.
- Bruk av flertråding[2] isteden for flere parallelle prosseser → mindre overhead[3] i operativsystemet.

[2] En tråd er en «lettvektsprosess» som deler adresseområde med en eller flere andre tråder, se ordliste for lengre forklaring.

[3] «administrasjonskostnader»

- Bedre kontroll over ressursforbruk og timing → Bedre kvalitet på videoleveranse.

Det er i fremste rekke Elvira2 som det er interessant å legge til båndlagring for, siden det er denne videotjeneren som vil bli satsingområde for senere utvikling. Arbeidet er i startfasen, og skriftlig informasjon er lite tilgjengelig. Derfor vil informasjonen om Elvira2 i denne rapporten være basert på muntlig kommunikasjon med Sandstå. Flere detaljer om Elvira2 finnes i avsnitt 5.1, side 45.

1.4 Valg av utstyr, verktøy og metoder

«Godt redskap er halve arbeidet» sies det. Dette gjelder i høyeste grad for utvikling av programvare og datasystemer. Her beskrives valg av utstyr for oppgaven. En del av valgene er gjort av andre, siden ESM skal integreres sterkt med programmer som allerede er implementert.

1.4.1 Konstruksjonsmetoder

Objektorientert konstruksjon har mange fordeler, særlig når det gjelder abstraksjon av problemer i flere delproblemer. Object Method Technology (OMT) (Rumbaugh et al. 1991) er en mye brukt metode for konstruksjon av objektorientert programvare. Designet av ESM vil bli presentert i OMT, i den grad det er hensiktsmessig. OMT består av et grafisk «bokspråk» og en rekke regler for hvordan man bør gå frem under konstruksjonen, og hvordan man overfører konstruksjonen til implementasjon.

Siden ESM vil få en forholdsvis moderat størrelse, i tillegg til at den må være optimalisert for høye overføringsrater, vil dette føre til at visse unntak fra en ren objektorientert konstruksjon vil bli foretatt. Dette gjelder i hvertfall på lavt nivå, der prosedyrale metoder vil bli brukt.

Databasebeskrivelser vil bli gjort i ER-diagrammer[4] (Elmasri og Navathe 1994). ER-diagrammer (og utvidelsen EER, der den første E-en står for Extended) er en mye brukt metode for modellering og konstruksjon av databaser. Slike diagrammer kan lett oversettes til implementasjonsnære beskrivelser for forskjellige typer databaser, slik som relasjons- og nettverksdatabaser.

1.4.2 Implementasjonsverktøy

Dagens Elvira er implementert i C++. Selv om C++ har sine ulemper, særlig når det gjelder kontroll med minnebruk, så hjelper programmer som Purify godt. Dessuten vil det være mulighet for gjenbruk av visse moduler, noe som ble vist under arbeidet med Elvira Catalog Manager (Sørensen og Sandstå 1996). Veileder uttrykte dessuten et ønske om at systemet skulle implementert i C++, og dette bør taes hensyn til, siden det er han som skal jobbe med systemet senere. C++ er derfor valgt som implementasjonsspråk for ESM. Det er aktuelt å bruke andre språk, f.eks. Java for å lage grafiske grensesnitt.

Her er en liste over verktøy/programvare som benyttes:

- Solaris 2.x – et operativsystem levert av Sun Microsystems, Inc. Solaris er et av flere operativsystemer som går under betegnelsen UNIX.
- Make – automatisert kompilering av programkode og lignende.
- Standard Template Library (STL) – et standardbibliotek for C++
- G++ – GNU C++, en gratis kompilator fra Free Software Foundation.
- Purify – sikkerhetssele for C++. Kommersiell programvare kjøpt inn av IDI. Analysator for minneaksesser.

[4] ER = Entity Relationship

- GDBM – enkel database som finnes på de fleste UNIX-systemer, eller med enkelhet kan installeres dersom den mangler. GDBM er implementert som et bibliotek som man lenker inn i C/C++ koden, på samme måte som f.eks. systembiblioteker.
- DOC++ – generering av dokumentasjon fra C/C++ headerfiler. Gratis programvare.
- RCS eller CVS – versjonskontroll (CVS er mer avansert, og tillater flere å jobbe parallelt med samme prosjekt). Gratis programvare.
- LaTeX2e – for å formattere denne rapporten.
- gnuplot – gratis program for plotting av data.
- perl – gratis programintepreter for scriptspråket Perl.

1.4.3 Tilgjengelig maskinvare

- 1 Tandberg TDC 4220 2.5 GB båndstasjon med tilhørende databånd.
- 2 Tandberg MLR-1 13 GB båndstasjoner med tilhørende databånd[5].
- Flere Axil 320 HyperSPARC-maskiner som kjører Solaris 2.4 (SunOS 5.4)
- En FORE ASX-200 ATM-svitsj.
- En Sun SPARC station IPX med en Tandberg MLR-1 båndstasjon.
- Diverse arbeidsstasjoner for programmering og dokumentasjon.

1.5 Informasjon for leseren

1.5.1 Valg av språk

Denne hovedoppgaven er primært skrevet på norsk. Figurtekster er skrevet på engelsk, fordi det er planer om å bruke dem i et engelsk dokument senere. Systemdokumentasjonen er skrevet på engelsk, fordi resten av Elvira er kodet og dokumentert på engelsk.

Norske betegnelser på data-spesifikke ord og uttrykk er brukt der hvor det er mulig. Dessverre har det ikke vært mulig å oversette alle ting til norsk, som «headerfil», «overhead», osv. I tillegg er det brukt engelske navn der det refereres til entiteter i implementasjonen.

1.5.2 Bruk av tall, priser og lignende

Det er en norsk tåpelighet å bruke komma som desimalskilletegn. Derfor vil alle desimaltall i denne hovedoppgaven bruke punktum som desimalskilletegn.

Alle priser er regnet uten merverdiavgift, fordi endel land ikke har dette, noe som vanskeliggjør konvertering til vedkommende lands valuta. Dessverre var det umulig å hente inn alle priseksempler på samme tid under arbeidet, så det kan være visse feil i kostnadsforskjeller. Markedet for datalagring har vært ekstremt ustabilt de siste årene.

1.5.3 Organisering av teksten

Merk at mange uttrykk og forkortelser er forklart i ordforklaringen på side 117. Resten av denne rapporten er organisert som følger:

Kapittel 2 Problemstilling – Problemanalyse og oppgavespesifikasjon.

[5]Disse ankom noe for sent i forhold til denne hovedoppgaven, slik at bakgrunnsmaterialet for modelleringen er basert på Tandberg TDC 4220.

Kapittel 3 Forstudium – Resultater av lesing av vitenskapelige artikler, rapporter og materiale funnet på Internettet.

Kapittel 4 Modell av et digitalt videoarkivsystem – Identifisering av parametre og skisse av modeller som kan beskrive et digitalt videoarkiv.

Kapittel 5 Problemer og løsninger – Løsning på problemer spesifisert i kapittel 2 og en evaluering av disse løsningene i lys av modellen og forstudiet.

Kapittel 6 Konstruksjon – Konstruksjonsbeskrivelse av ESM. Tildels overordnet.

Kapittel 7 Implementasjon – Beskrivelse av det som er implementert av ESM.

Kapittel 8 Målinger – Målinger gjort på delene ESM som ble ferdigstilt i implementasjonen.

Kapittel 9 Konklusjon – En oppsummering av det som har blitt konstruert, implementert, modellert og målt. Slutninger trukket av arbeidet med ESM.

Tillegg A Ordliste – Ordliste med forklaringer av forkortelser og spesielle ord.

Tillegg B Møte med NRK – Referat av møte med NRK 1996-12-13.

Tillegg C Beregninger og illustrasjoner – Illustrerende eksempler med tall tatt fra både luften, virkeligheten og teorien.

Tillegg D Systemdokumentasjon – Detaljerte beskrivelser av alle kodegrensesnitt i ESM.

Eget vedlegg Kildekode – Kildekode for ESM og støttemoduler.

1.5.4 Begreper brukt i forbindelse med lagring av data

Endel begreper relatert til lagring av data går igjen i denne rapporten. Disse trenger en kort forklaring, ut over det som står i ordlisten på side 117.

1.5.4.1 Noen ordforklaringer

Overføringsrate er den hastigheten man klarer å overføre data mellom to enheter med. *Aksesstid* er tiden det tar fra man gjør en forespørsel om bestemte data, til første byte av disse dataene har ankommet. *Overføringstid* er tiden det tar å overføre alle dataene mellom to punkter, og er avhengig av aksesstid, overføringsrate og mengden data som skal overføres.

1.5.4.2 Lagerhierarkier og klassifisering

Lagerhierarkier er en løsning for å få «det beste av to verdener» med hensyn på lagringsmediers hastighet og pris (Silberschatz og Galvin 1994). På toppen av hierarkiet finner man raske og dyre løsninger. På bunnen finnes langsommere og billigere løsninger. Man deler gjerne inn lagerenheter i tre hovedgrupper: primær-, sekundær- og tertiær-lager. *Primærlager* brukes om lager av halvledertypen der aksesstiden er uavhengig av hvilken adresse de lagrede dataene har. Primærlager kan også omfatte registre og eventuelle hurtigbuffere (Silberschatz og Galvin 1994). *Sekundærlager* brukes gjerne om harddisker koplet til maskinen. Sekundærlager har rundt to størrelsesordner større lagerkapasitet, men med aksesstider som er rundt 5 størrelsesordener høyere.

Tertiærlager brukes om datalager hvor man har minst én størrelsorden høyere verdier enn sekundærlager for hver av disse egenskapene (Silberschatz og Hillyer 1996*c*):

- lagringskapasitet per enhet
- kostnadseffektivitet (f.eks. målt i bits per krone)

Nivå	Aksesstid	Aksessmønster	Kostnad	Typisk kapasitet (enkeltenhet)
primær	nanosekunder	tilfeldig	titalls kr/MB	32 MB
sekundær	millisekunder	direkte	få kr/MB	4 GB
tertiær	sekunder/minutter	sekvensielt	få øre/MB	20 GB

Tabell 1.1: Kort oppsummering av typiske egenskaper til nivåene i et tredelt lagerhierarki

- aksesstid

Båndstasjoner og spillere for optiske disker er det vanlig å klassifisere som tertiære lagringsmedier.

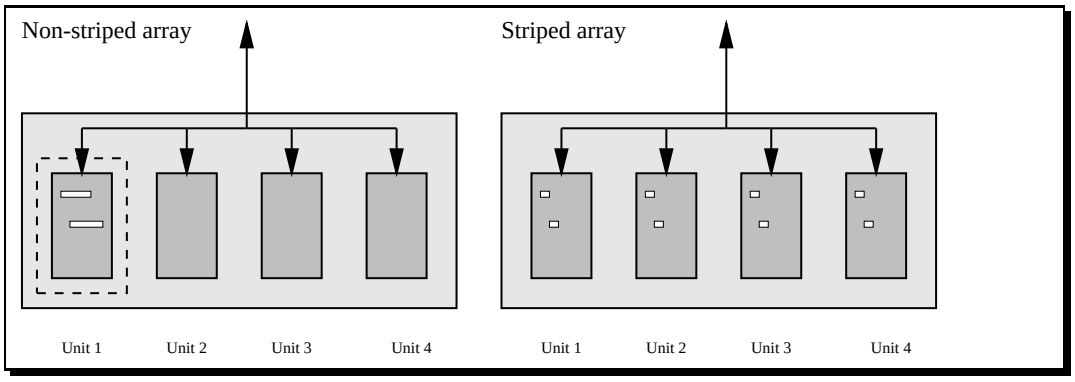
Inndelingen ovenfor er ment å være en overordnet inndeling. Innenfor hver lagertype kan det finnes flere nyanser. Tertiærlager kan for eksempel deles ytterligere opp i to grupper: enkeltstående enheter og biblioteksystemer. Primærlager kan fin-inndeles i registerminne, hurtigbuffer (opptil flere nivåer) og hovedminne.

Det finnes lagerprodukter på markedet i dag som ikke faller inn under noen av betegnelsene. Spesielle stasjoner som tilbyr utskiftbare harddisker (SyQuest Technology, Inc. 1996, Iomega Corporation 1996), der man typisk har utskiftbare harddisker som koster rundt 700 kroner stykket, finnes på markedet i dag. Spesifikasjonene for disse harddiskene er tilnærmet lik det man finner på vanlige stasjonære harddisker. Men strengt tatt er slike stasjoner sekundære kun når det står en fast disk i stasjonen. Dersom disken er ute av maskinen må det en manuell eller mekanisk handling til for å få den på plass i stasjonen. Da får enheten en tertiær karakteristikk, med hensyn på aksesstid.

En skjematisk fremstilling av forskjellene mellom de tre typene lagringsmedier med hensyn på aksesstid, aksessmønster, kostnad og kapasitet finnes i tabell 1.1.

1.5.4.3 Striping av data

Dersom man har mange lagerenheter i et stort system (f.eks. i en videotjener) så er det en viss sannsynlighet for at brukerne vil ha tak i visse filer oftere enn andre. Til venstre i figur 1.1 ser man et eksempel på et system med fire lagringsenheter og to filer på enhet 1 som blir aksessert svært ofte av brukerne. Siden flertallet av aksesser gjøres mot enhet 1, vil denne lett bli overbelastet, mens de andre enhetene får lite å gjøre. Enhet 1 blir en flaskehals i systemet. Dersom man fordeler datafilene over flere enheter, vil man få en lagring som skissert til høyre i figur 1.1. Enhver fil som lastes ned, vil da belaste alle enhetene, men i et kortere tidsintervall siden hver del er ganske liten i forhold til den ustripede filen. Dette fører til en utjevning av belastningen, slik at flaskehalsen forsvinner. (Ford et al. 1996, Chervenak et al. 1995, Bratbergsengen 1996).

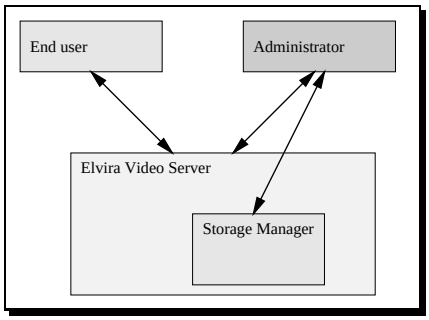


Figur 1.1: Striping i lagringsmedier

Striping av data er ikke gratis. Det må lages et kontrollsystem i hardware eller software som er mye mer komplekst enn kontrollsystemene man finner i enkeltenhetene. Som kapittel 4 vil vise, kan man også bli mer følsom for maskinvarefeil.

1.5.5 Brukerbegrepet

Det finnes to hovedgrupper brukere av Elvira. Administrator-brukere overvåker og vedlikeholder tjeneren. Sluttbrukere benytter Elvira til å se på og redigere kontinuerlige multimediadata. I denne rapporten vil sluttbrukerne heretter bare omtales som brukere, og administratorbrukere som administratorer.



Figur 1.2: Brukerne av Storage Manager

Begge brukergrupper vil indirekte bruke ESM når den er integrert med Elvira. I tillegg vil administratorene bruke funksjonalitet fra ESM direkte når det gjelder overvåkning, kontroll, sikkerhetskopiering og andre lignende aktiviteter. Dette er vist i figur 1.2. Dette betyr ikke at endringene i systemet vil være umerkelige for brukerne, for det vil de bli. Overgang fra lagring på sekundære media til lagring på tertiære media, må nødvendigvis få konsekvenser for brukerne. Det figuren forsøker å vise, er at ESM kun kommuniserer med andre moduler i Elvira og ESM-administratorer.

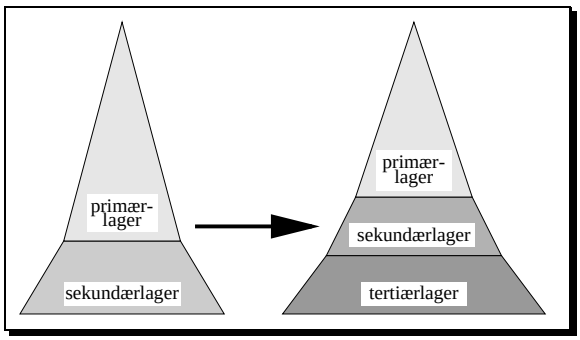
Kapittel 2

Problemstilling

I dag har IDI en videotjener, kalt Elvira Video Server (Sandstå et al. 1996, Langørgen 1994), som leverer digital video og audio (mediadata) over et høyhastighetsnettverk. Disse mediadataene blir lagret på flere harddisker koblet til en klynge av arbeidsstasjoner. For å kunne bygge opp store videodatabaser, som f.eks. et fjernsynsarkiv for NRK, er det nødvendig å lagre svært store mengder med digital video. Disse datamengdene vil kunne komme opp i flere terabyte. Å lagre all denne informasjonen på harddisk vil medføre altfor høye kostnader.

Løsningen er altså å utvide kapasiteten med lagringsmedier som er billigere enn harddisker, og som vil fortsette å være det i en viss tid fremover.[1]

Det er derfor aktuelt å introdusere ett nytt nivå i lagerhierarkiet til Elvira. Dagens system flytter data fra sekundærlager (harddisker) til primærlager (hovedminne), før de sendes ut over nettet. Etter utvidelsen vil vi få et større hierarki, der harddiskene også vil fungere som buffere mot et tertiærminne som, i hvertfall i første omgang, vil bestå av en eller flere båndstasjoner, og senere av automatiske båndbiblioteker. Figur 2.1 illustrerer utvidelsen av lagerhierarkiet, riktignok på en svært forenklet måte.

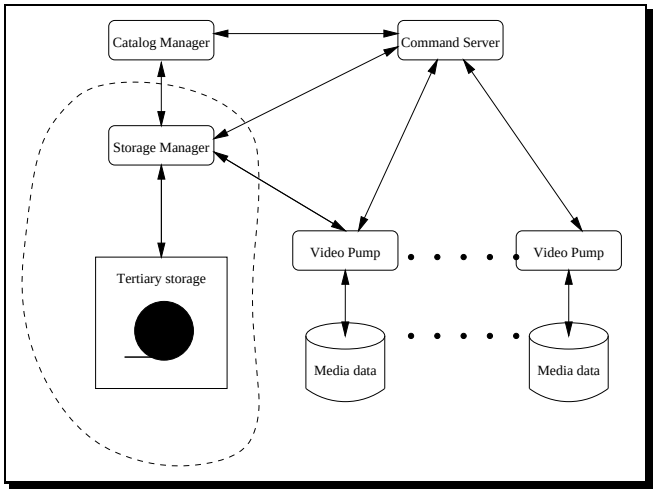


Figur 2.1: Overgangen fra hovedlager på sekundærnivå til tertiærnivå

Når man går fra primærlager til sekundærlager opplever man at aksesstiden til dataene går fra noen titalls nanosekunder til rundt ti millisekunder, altså en økning på 5 størrelsesordner. Ved overgang fra sekundærlager til et typisk tertiærlager som f.eks. bånd, vil man oppleve en økning i aksesstiden på ytterligere 3-4 størrelsesordner, slik at man må vente i flere titalls sekunder (Silberschatz og Hillyer 1996*a*).

Den planlagte utvidelsen av Elvira er illustrert i figur 2.2, og er merket med en stiplet linje. En konseptuell skisse for hvordan et lagerhierarki for et stort arkivsystem vil se ut er vist i figur 2.3.

[1] Prisen på harddisker i forhold til båndstasjoner er i dag unaturlig lav, fordi det blir produsert enorme mengder



Figur 2.2: Høynivå prosessdiagram av Elvira2 med ESM

Det forestående arbeidet kan deles opp i tre hoveddeler:

- Bruk av båndstasjon i Elvira må modelleres.
- Det må konstrueres og implementeres et programsystem som flytter data mellom bånd og harddisker (en Storage Manager), og Elvira Video Server må modifiseres for bruk av denne enheten.
- På grunnlag av modeller og målinger skal Elvira med båndlagring utredes og evalueres.

2.1 Modellering

Man kan ikke gå fra lagring på harddisker til et tertiært lagringsmedium uten at det får konsekvenser med hensyn på aksesstid og andre parametre. Ved å modellere oppførselen til ESM på flere nivåer får man:

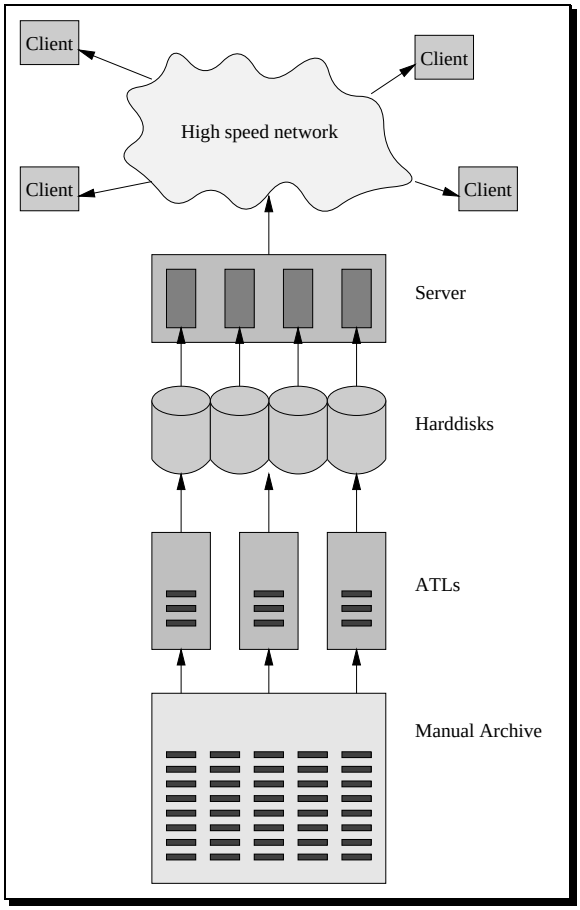
- mulighet for å sammenligne teoretiske antagelser om oppførsel mot praktiske målinger.
- en referanseramme for diskusjoner om effektivitet, ønsket oppførsel, osv.
- en nyttig nedbrytning av et forholdsvis komplekst problem, slik at det blir enklere å sette seg inn i det.

2.2 Utvikling av lagerhåndterer

Det må utvikles en modul i videotjeneren som skal ta seg av flytting av data mellom båndstasjon og harddisker. Deler av videotjeneren må kanskje endres slik at de kan benytte seg av båndlagring. Videre må det lages et system som håndterer brukernes forespørsler om henting av film, og til slutt må det lages et system som passer på at harddiskene ikke blir fulle.

Oppgaven består i å konstruere og implementere denne modulen, som vil bli kalt Elvira Storage Manager. I tillegg må koden instrumenteres slik at målingene beskrevet i neste avsnitt kan gjennomføres.

harddisker i forhold til båndstasjoner. Masseproduksjon har som kjent en tendens til å senke prisen per enhet.



Figur 2.3: Konseptuell skisse av lagerhierarki i et stort arkiv for digital video

2.3 Målinger og evaluering

Når Storage Manager er konstruert og implementert vil det være viktig å analysere hvordan båndlagring påvirker resten av videotjeneren.

Analysen består i å ha en modell som forutsier visse verdier for systemet og deretter foreta målinger for å sjekke om målte verdier er i samsvar med de predikerte. Dette bør gjøres i flere omganger, slik at man får frem en modell som med høy sannsynlighet er representativ.

Det er viktig å finne ut hvordan sluttbrukerne av Elvira Video Server vil merke endringene som er gjort. Det vil også være aktuelt å bruke dataene til å justere visse parametre i Storage Manager for å øke effektiviteten.

Kapittel 3

Forstudium

I dette kapittelet diskuteres først forskjellige typer båndteknologi og annen masselagerteknologi. Deretter presenteres forskjellige videotjenere, forskjellige formater for video, og til slutt presenteres endel av forskningsarbeidet som foregår innen dette fagfeltet.

3.1 Masselagersystemer

Med masselagersystemer, mener vi systemer som tilbyr persistent[1] lagring av store mengder data, gjerne mer enn én terabyte.

Når man ser på masselagersystemer er kostnad en av de viktigste egenskapene. Kostnadene ved bruk av masselagersystemer kan deles inn i to komponenter: kostnader for maskinvaren og kostnader for det utskiftbare lagringsmediet[2]. Det er selvsagt også andre kostnader knyttet til masselagersystemer, slik som vedlikeholdskostnader og generelle driftskostnader.

Informasjonen fra produsentene er ofte overdrevne, og med hensikt gjort ganske uklare. Når det gjelder tall på kapasitet og overføringsrater i denne delen, så er det alltid snakk om lagring av *ukomprimerte data*. Priser er oppgitt *eksklusive merverdiavgift* eller andre former for omsetningsavgift.

3.1.1 Båndteknologi

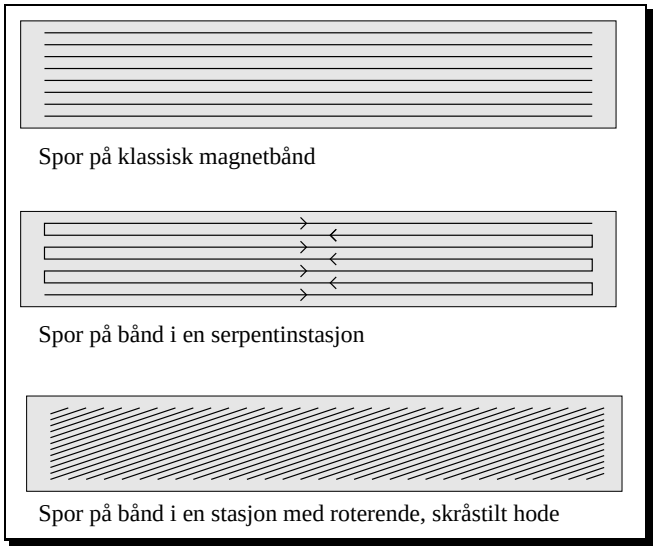
Det finnes flere typer av båndteknologier. Disse er utførlig beskrevet av blant annet Bratbergsengen (1996), Silberschatz og Hillyer (1996*c*) og Saha (1996). I denne delen beskrives de vanligste teknikkene i forbindelse med lagring på magnetiske bånd.

Følgende generelle egenskaper skiller lagring på bånd fra lagring på harddisk:

- Bånd er et *sekvensielt* medium, fordi båndets natur krever at man må lese datablokk for datablokk for å finne den blokken man er interessert i. Dette fører til store aksesstider i forhold til disker, der man kan aksessere en blokk direkte.
- Bånd er montert i utskiftbare kassetter, som monteres i båndstasjoner. Harddisker er (stort sett) fastmontert i stasjonen. Dette gjør at kostnadskurvene for bruk av harddisk og båndstasjon ser svært forskjellige ut, slik som skissert i figur 3.2.

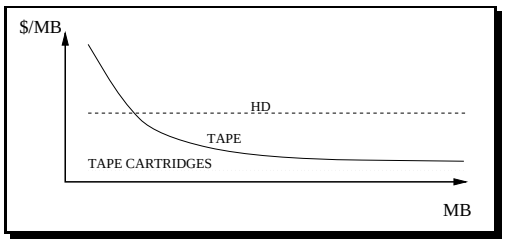
[1] persistent = varig, holdbar

[2] Dette gjelder selvsagt ikke vanlige harddisker, der mediet er fastmontert i stasjonen og ikke kan tas ut uten å ødelegge den.



Figur 3.1: Spormønstre på magnetbånd (Bratbergsengen 1996)

- Siden bånd stort sett vil være off-line i et stort arkiv, innebærer dette at mediet ikke er i mekanisk bevegelse annet enn i en liten brøkdel av levetiden sin. Harddisker er derimot aktive hele tiden. Dette bør taes hensyn til når man regner på feilrater.



Figur 3.2: Skisse av kostnadsutvikling ved kapasitetsøkning

Figur 3.2 er en skisse for kostnadsutvikling for en båndstasjon, versus flere harddisker. Her er det tenkt at man kjøper inn f.eks. 12 og 12 gigabyte av gangen. Initielt vil en båndstasjon være mye dyrere per megabyte enn en harddisk, siden båndstasjoner er svært kostbare. Hver gang man skal utvide med harddisk, må man kjøpe en ny, så denne kostnaden blir konstant. Ved utviding med bånd, trenger man bare å kjøpe en ny kassett, som er mye billigere enn en harddisk. Derfor vil NOK/MB konvergere mot kostnadseffektiviteten til båndkassetter alene. På dette området slår lagring på bånd knockout på harddiskbasert lagring med en til to størrelsesordner lavere kostnad per megabyte. Merk at dersom man tar med overføringshastigheter og/eller aksesstider, får man et helt annet bilde av kostnader. Dette vil diskuteres senere.

I arkivsammenheng er det svært viktig at feilraten er så liten som mulig, siden man ellers vil få en enorm jobb med å stadig vekk kopiere gamle bånd over til nye.

3.1.1.1 Klassiske magnetbånd

Disse typene magnetbånd er ofte en halv tomme brede. Bitene lagres på tvers av båndet i såkalte rammer (frames), med ekstra biter for paritetssjekk. Spormønsteret er skissert i figur 3.1. Denne typen bånd ble brukt i stormaskinsystemer, stasjonene var gjerne enorme, og mekanikken som ble

brukt var intrikat, for å si det mildt. Teknikken er videreført i IBMs båndstasjoner 3480, 3490 og 3490E (Bratbergsengen 1996). 3490E har omtrent 800MB lagringskapasitet og overføringsrate på rundt 3 MB/s (Van Meter og Stith 1996).

3.1.1.2 Bånd med lagring i serpentinmønster

Lineære serpentinbånd har flere parallelle spor der databitene lagres sekvensielt. Sporene er ordnet i et serpentinmønster, som vist i figur 3.1.

Det finnes flere formater for bånd der man lagrer data i serpentinmønster. Digital Linear Tape (DLT), Quarter Inch Cassette (QIC) og Travan er de mest kjente. DLT er et produktnavn, mens QIC og Travan er et standard kassettformat som egentlig ikke legger noen føringer for hvordan en stasjon kan lagre data på båndet. Prisene for en DLT 4000 med kapasitet på 20 GB ligger i området 32-37000NOK (Van Meter og Stith 1996, Wave Technologies 1997).

DLT-bånd inneholder kun en spole. Båndstasjonen må selv sørge for å spole opp båndet den leser fra kassetten, og når kassetten skal demonteres, må man selvsagt spole alt båndet tilbake til kassetten. Den siste generasjon DLT-stasjoner heter DLT 7000 (Quantum Corporation 1997, Van Meter og Stith 1996) og kan lagre 35 GB per bånd og har en kontinuerlig overføringsrate på 5.0 MB/s.

QIC-kassetter inneholder to spoler, i motsetning til DLT. Båndet er halvparten så bredt, og man har dermed ikke like stor lagringskapasitet per lengdeenhet som DLT. (Dette er jo i utgangspunktet en urettferdig sammenligning). Fordelen med QIC er at kassettene er *svært* robuste, og at man slipper å spole opp båndet internt i stasjonen, noe som fører til at båndet er mer isolert fra stasjonen. QIC startet ut som et billig alternativ til dyrere teknologi, men QIC har nærmet seg DLT både i kapasitet og pris. Prisen for en 13GB MLR-1 fra Tandberg Data er på rundt 30000NOK.

Travan-kassetter er en oppgradering QIC. Teknologiene er nesten identiske, bortsett fra at båndet er 0.315 tommer isteden for 0.250 tommer. Travan-stasjoner skal kunne lese det smalere QIC-formatet (Van Meter og Stith 1996, Silberschatz og Hillyer 1996*c*).

3.1.1.3 Båndstasjoner med roterende skråstilte hoder

Disse stasjonene har en eller flere hoder montert på en sylindrisk skanner som roterer på skrått i forhold til båndet som føres forholdsvis sakte forbi. En illustrasjon av mønsteret er gitt i figur 3.1.

Det eksisterer flere typer båndstasjoner som baserer seg på teknologien med roterende skråstilte hoder. Formatet på båndet er gjerne 4mm (DAT), 8mm (video) eller gode gamle VHS-kassetter. Disse båndstasjonene har gjerne svært stor kapasitet, og de variable kostnadene (dvs. kostnad for båndkassetter) er blant de laveste.

Utnyttelsen av båndet er forholdsvis høy. All magnetisk lagring er basert på at det induseres strøm i et hode som beveger seg igjennom vekslende magnetfelter. Skanneren med lese/skrivehodene roterer raskt, og fører til at hodenes relative hastighet til båndet blir høy, noe som muliggjør lagring med stor tetthet.

Den største ulempen med denne typen lagring er at det roterende hodet (skanneren) sliter uforholdsmessig mye på båndene, slik at de blir slitt fortere ut enn bånd som brukes i serpentinstasjoner. Faktisk sliter hodet så mye på båndet at dersom båndstasjonen stopper opp, må båndet avmonteres fra den roterende trommelen ganske fort, så båndet ikke slites av. Dette fører til en liten forsinkelse når man skal begynne å lese båndet igjen (Bratbergsengen 1996). Et typisk bånd blir slitt ut etter bare noen få tusen passeringer.

Det finnes flere produsenter av båndstasjoner som benytter roterende skråstilte hoder. Exabyte og Sony er to av de største aktørene på dette markedet. Exabyte leverer en stasjon kalt Mammoth eller EXB-890. Denne har en kapasitet på 20GB og en overføringsrate på 3MB/s (Exabyte Corporation 1997).

3.1.2 Andre typer masselager

Det finnes flere forskjellige andre systemer for lagring av store mengder data som akkurat nå har større kostnad per byte enn bånd, men som i fremtiden kan vise seg å være mer kostnadseffektive.

3.1.2.1 Optiske og magnetooptiske disker

En teknologi som blir mye brukt på forbrukermarkedet er lagring av data på optiske disker. Hovedproblemet med slike er at de vanligvis har forholdsvis liten kapasitet (rundt 500MB), samt at overføringsraten er ganske lav.

ROM-disker (Read Only) er optiske disker der dataene legges inn under produksjon i en slags trykkeriprosess. For store mengder identiske kopier, blir dette en svært kostnadseffektiv prosess. Det er få ting som slår CD-ROM når det gjelder global distribusjon av programvare og store mengder data.

WORM-disker (Write Once, Read Many) disker har eksistert på markedet en stund. WORM er ikke en teknologi, men heller en beskrivelse à la ROM. WORM CD-ROM plater kan fåes kjøpt i butikken for rundt 70 kroner her i Norge. Grunnen til at man kun kan skrive en plate en gang, er at «brenneren» lager små hull i mediet ved hjelp av en intensiv laser. I arkivsammenheng kan bruk av WORM-medier være interessant, særlig når man tenker på permanent lagring. Dessverre har ikke f.eks. CD-plater ubegrenset holdbarhetstid, og må i likhet med bånd oppfriskes en gang i blant. Det er ganske stor uenighet om livslengden til CD-plater, men 10-14 år har vært nevnt. Når man skal friske opp en WORM-plate, må man overføre dataene til nye plater. Sony selger en WORM-stasjon kalt WDD-531 for 12-tommers plater som har kapasitet på 15GB med overføringsrater på mellom 0.6 og 1.3 MB/s (Sony Corporation 1997).

Magnetooptiske disker er plastikk- eller glass-disker med et belegg med spesielle egenskaper. Disken blir lest ved å rette en laserstråle av lav intensitet mot mediet og deretter tolke polariseringen til det reflekterte lyset. Skriving foregår ved å bruke en laser med høyere intensitet som varmer opp belegget og dermed gjør det mer mottakelig for magnetisk påvirkning fra det magnetiske skrivehodet. Et eksempel på denne teknologien er Sony MiniDisc, som i datalagrings-versjon kan lagre inntil 140MB (Van Meter og Stith 1996). Magnetooptiske disker kan i motsetning til WORM-disker skrives flere ganger.

Digital Versatile Disc (Bell 1996), vanligvis forkortet til DVD, er et nytt format for optiske disker. Disse diskene vil være en stor forbedring i forhold til dagens CD-ROM-teknologi. De første DVD-platene vil ha en kapasitet på 9.4 GB[3], men det er planlagt en dobling av denne kapasiteten på senere versjoner, slik at man vil kunne lagre opptil 17GB. For å få til denne doblingen, vil man lagre data på to dybdenivåer. De første DVD-ene som kommer vil være av ROM-typen. Etterhvert vil WORM-versjoner komme. En versjon som kan skrives flere ganger, kalt DVD-R(ecordable), vil være aktuell som lagermedium i multimediasystemer[4].

3.1.2.2 Optiske bånd

Optiske båndstasjoner er antakeligvis meget interessante kandidater til bruk i digitale videoarkiver. Som man kan se av tabell 3.1, side 19 så er optiske båndstasjoner svært dyre akkurat nå, men dette skyldes i hovedsak at de ikke har kommet i masseproduksjon ennå.

Et kanadisk firma med navn CREO har laget en optisk båndstasjon som kan lagre 1 TB på hvert bånd, som er 800 meter lange, dvs. omtrent samme lengde som på vanlige magnetiske bånd. Siden overføringsraten ikke er høyere enn 3 MB/s, betyr det at å lese et bånd (kontinuerlig) tar fire dager! (Bratbergsengen 1996, Van Meter og Stith 1996). Det er rapportert at denne båndstasjonen koster godt over en million NOK og at hvert bånd koster over 40000NOK, noe som gjør teknologien

[3] 4.7 GB på hver side.
[4] ROM DVD er akkurat nå i ferd med å dukke opp, i følge USENET-gruppen no.radio-tv.

uinteressant for alle andre enn svært store organisasjoner og institusjoner. Båndene skal ha en levetid på rundt 15 år.

3.1.2.3 Harddisker som masselagre

Prisen på harddisker har sunket såpass den siste tiden at de utgjør et seriøst alternativ til bånd som hovedlagringsmedium for store mediadatabaser.

Bruk av harddisker gir flere store fordeler, jf. tabell 3.1, side 19. På en harddisk vil ikke lese/skrive-hodene berøre mediet, men flyte like over det. Dette gjør at en harddisk er mer motstandsdyktig mot slitasje enn et bånd som er i direkte kontakt med lese/skrive-hodet. I tillegg har harddisker en vesentlig kortere aksesstid. Striping (se avsnitt 1.2, side 2) av data over flere disker (f.eks. i en RAID-konfigurasjon) vil også gi en mye større båndbredde på I/U enn det en enkelt vanlig båndstasjon kan klare.

Det er også trolig at man måtte brukt en eller annen form for backup i et slikt harddiskbasert system, eller benytte seg av et RAID-nivå som muliggjør rekonstruering av tapte disker.

Det finnes forskere som påstår at kostnadene per megabyte for «hjemmesnekrede» RAIDs kan bli lavere enn for automatiske båndbiblioteker allerede i 1997, se mer om dette i avsnitt 3.5.4, side 27.

Seagate leverer en harddisk i 5.25-tommers format som har en kapasitet på hele 23.2GB. Den har en overføringsrate på rundt 20MB/s, lover produsenten (Seagate 1997). Prisen for denne disken ligger på mellom 29000 og 34000NOK[5] (Tech Computers 1997). Dette gir en pris per megabyte på rundt 1.25 NOK.

3.1.3 Biblioteksystemer

Et biblioteksystem består av en samling med enkeltmedier (båndkassetter, optiske disker, etc.) og en eller flere stasjoner for lesing og skriving av disse mediene. Store biblioteker gjør at man kan avskrive kostnadene for anskaffelse av forholdsvis dyre stasjoner over svært store datamengder, slik at gjennomsnittskostnaden nærmer seg den for det utskiftbare enkeltmediet, slik som vist i figur 3.2, side 14.

Manuelle biblioteksystemer består av hylleplass for plassering av bånd og en eller flere ansatte som har som oppgave å flytte båndkassetter (eller andre former for lageringsenheter, som f.eks. disker) mellom et arkiv og en eller flere stasjoner for lesing/skriving. Slike lagersystemer blir ofte betegnet som *off-line*. Store manuelle biblioteksystemer kan ha tusenvis av bånd lagret i en eller flere klimakontrollerte, brannsikrede rom. Slike biblioteker er ikke noe man kjøper fra produsenter/leverandører, men heller bygger opp med egne ressurser.

Automatiske biblioteksystemer består av en eller flere stasjoner og en «robot» som forer stasjonene med enkeltmedier. Man bruker ofte betegnelsen *near-line* om automatiske biblioteksystemer. Slike automater kan ha en ganske høy anskaffelseskostnad, men det er viktig å huske på at i manuelle systemer vil utgifter til lønning av arkivarbeidere være forholdsvis store per år. Tiden det tar å bytte ut et medium med et annet i en stasjon er svært avhengig av hvor stort arkivet er (dvs. hvor store geografiske avstander roboten skal bevege seg over), og hva slags type robot man har. Utskiftningstiden kan være på fra noen få sekunder til flere minutter.

Dersom man har enormt store datamengder som skal lagres, kan det være lønnsomt med en kombinasjon, der arkivarbeidere skifter ut båndkassetter mellom et stort arkiv og en eller flere automater.

[5] Prisforskjell mellom to dager!

3.1.3.1 Automatiske båndbiblioteker

Båndbiblioteker blir ofte betegnet som ATL-er (Automatic Tape Library). Båndbiblioteker introduserer i hovedsak disse «forverringene» i forhold til en båndstasjon med kun ett bånd:

- Tiden det tar å fjerne båndet som allerede står i stasjonen, og eventuelt spole det tilbake.
- Tiden det tar å plassere det gamle båndet på sin lagerplass.
- Tiden det tar å finne fram et bånd fra lagerplassen i biblioteket og bringe den til båndstasjonen.
- Tiden det tar for båndstasjonen å montere båndet.

På den andre siden har gjerne båndbibliotekautomater mer enn en båndstasjon installert, slik at flere brukere kan betjenes på en gang. Det finnes flere forskjellige typer båndbiblioteker. De går under betegnelser som: stacker, jukeboks, karusell og silo (Silberschatz og Hillyer 1996*c*). Disse betegnelsene er først og fremst en beskrivelse av den mekaniske oppbygningen av systemet, og ikke særlig relevant i denne sammenheng.

Tandberg Data Storage leverer et bibliotek kalt TDS 1440 som har en kapasitet på 520GB og inneholder opptil 40 kassetter. Den kan ha mellom 1 og 4 MLR-1 stasjoner for lesing og skriving. Det tar ca. 10 sekunder å laste et bånd i en stasjon, gitt at den er ledig (Tandberg Data 1996). For en TDC 1440 med 2 MLR-1 båndstasjoner er prisen rundt 400kNOK.

Exabyte leverer et bibliotek som har kapasitet på 1.6TB (Exabyte Corporation 1997), som tar 80 bånd. Den har 4 Mammoth-stasjoner for lesing og skriving. Denne bruker også rundt 10 sekunder på å laste et bånd i en stasjon.

En DLT jukeboks med 264 bånd og 3 båndstasjoner koster rundt 800kNOK (Wave Technologies 1997). Dersom man bruker 20GB-stasjoner, er man i stand til å lagre over 5 terabyte i et bibliotek.

3.1.3.2 Andre typer biblioteker

Det finnes også automatiske biblioteker for optiske disker. Disse har noenlunde samme egenskaper som båndbibliotekene, men forsinkelsene i forbindelse med innsetting og fjerning av medier fra stasjonene er ofte mindre.

Sony selger en jukeboks for WORM-plater kalt WDA-E550, og den har en kapasitet på 1.14TB (Sony Corporation 1997). Denne veier over 300kg i basiskonfigurasjon og inneholder minst 76 plater. Den kan også utvides ytterligere. Tiden det tar å mate en stasjon med en plate ligger på mellom 6 og 10 sekunder (avhengig av hvor mange drev som står i stasjonen).

3.1.4 Annen bruk av tertiære medier

Nå for tiden er tertiære medier i hovedsak brukt til andre formål, blant annet til:

- Sikkerhetskopiering
- Dataintensive målinger innen medisin, astronomi og geologi.
- Programvaredistribusjon

Bruken av tertiære lagringsmedier i sikkerhetskopi-sammenheng er ikke særlig relevant for denne oppgaven, siden sikkerhetskopiering stort sett går ut på å lagre data på bånd for å forhåpentligvis aldri senere hente det ut igjen.

I forbindelse med astronomi, er det gjerne slik at man må hente ned virkelig store mengder data med høy overføringsrate. Dette materialet blir siden batch-prosessert (Bratbergsengen 1996). Det

er altså ikke snakk om å bruke materialet aktivt når det ligger på bånd. Dette er nå i ferd med å endre seg, slik som beskrevet i avsnitt 3.5.3, side 27. Innen medisin trenger man ofte å lagre store mengder data i forbindelse med f.eks. MR- og CT-skanning av pasienter. Ved geologiske undersøkelser, gjerne i forbindelse med leting etter olje, genereres store mengder data i forbindelse med seismologiske undersøkelser.

Store programsystemer på flere hundretalls megabyte blir ofte distribuert på CD-ROM eller bånd. Overføring via nettverk er rett og slett ikke kostnadseffektivt for distribuering av så store mengder data til så mange kunder[6].

3.1.5 Oppsummering og konklusjoner

I tabell 3.1 presenteres omtrentelige verdier for visse interessante egenskaper ved forskjellige typer og systemer masselagring av data.

Navn	Overføringsrate	Variabel kostnad	Fast kostnad	Kapasitet[7]	Gj.snitt aksesstid	Pålitelighet
Harddisk	10MB/s	1.25NOK/MB	—	23.2GB	noen msek	høy
Serpentin	3.0MB/s	0.02NOK/MB	30kNOK	35GB	titalls sek	høy
Roterende hoder	3.0MB/s	0.01NOK/MB	30kNOK	>20GB	titalls sek	middels/lav
Optiske bånd	3.0MB/s	0.05NOK/MB	5000kNOK	1000GB	titalls sek	høy
Optisk disk	600kB/s	0.4NOK/MB	...	10GB	få sek	høy

Tabell 3.1: Oppsummering av masselagersystemer

Konklusjonen som man kan trekke på grunnlag av dette avsnittet er at at ESM ikke bør gjøre seg avhengig av noen spesiell teknologi for lagring av video og audio. Det vil hele tiden komme nye medier med forskjellige karakteristika. For å gjøre ESM uavhengig av lagringsmedium, kreves det en modulær og lagdelt oppbygging, slik at de høyere nivåer ikke er avhengig av den underliggende maskinvaren. Utviklingen innen tertiære medier har på ingen måte stoppet opp, og derfor er det viktig at kun små endringer må gjøres dersom maskinvaren skal oppgraderes.

3.2 Andre videotjenere og videoarkiv

De aller fleste videotjenere som i dag er implementert eller under utvikling er basert på lagring av mediadata på harddisk. Hovedparten av videotjenere er dessuten optimalisert med tanke på Video-On-Demand (VOD). Et viktig moment er at i en arkivsammenheng vil ikke VOD-arkitektur være kostnadsoptimalt, siden det er snakk om å lagre mye større mengder data per bruker. Det er gjort endel illustrerende beregninger på dette i tillegg C. Først beskrives fire videotjenere fra store kommersielle leverandører, og deretter et system fra University of California, Berkeley.

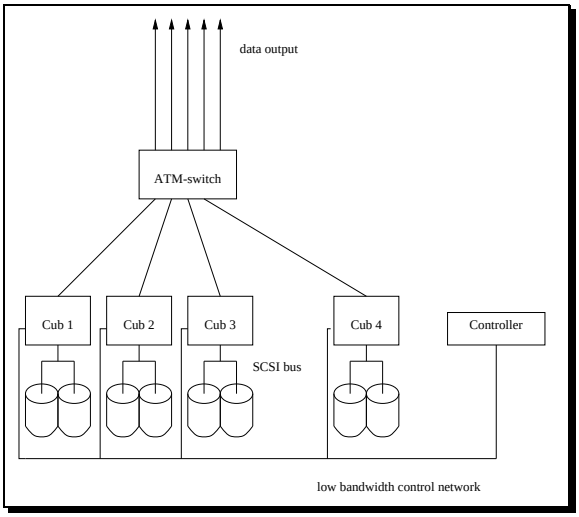
3.2.1 Microsoft Tiger

Microsoft har laget en videotjener kalt Tiger (Bolosky et al. 1996) basert på arbeidsstasjoner koblet sammen via en ATM-svitsj. Harddisker er hovedmedium for lagring av data. Dataene stripes (se avsnitt 1.2, side 2) over alle nodene. Dette gjøres for å oppnå en jevn fordeling av last på nodene. Hver harddisk er koplet til en arbeidsstasjon. Nodene er koplet sammen via en noe spesielt konfigurert ATM-svitsj, der man har en slags alle-til-en krets. Dette er skissert i figur 3.3.

For å minske kostnadene, har man brukt maskinvare som selges i store volumer. Det finnes endel begrensninger i Tiger. Blant annet må all video lagres med samme båndbredde i en tjener. Det har

[6] «Never underestimate the bandwidth of a station wagon full of tapes.» – Dr. Warren Jackson, Director, UTCS, University Of Texas

[7] Største kjente kapasitet for enkeltmedium



Figur 3.3: Oppsett av Tiger (Bolosky et al. 1996)

blitt kjørt endel tester av Tiger, der man har lagret alle data med båndbredder fra 1.5 Mbit/s til 8.0 Mbit/s.

3.2.2 Oracle Media Server

Oracle Media Server (Laursen et al. 1994, Oracle Corporation 1996) er en av tre software-systemer som Oracle tilbyr i forbindelse med interaktive TV-tjenester. De to andre komponentene kalles Oracle Media Net og Oracle Media Objects.

Media Server består av tre typer datalagringsmoduler, som er spesialisert for forskjellige typer data. Kontinuerlige datastrømmer (audio og video) lagres i Media Data Store, et filsystem som striper dataene over flere harddisker i softwareimplementert RAID. Tekstlige data er lagret i en spesiell tekstdatabase med ganske avanserte søkemuligheter, og strukturerte data lagres i Oracle RDBMS[8]. Tjeneren kan kjøres på flere typer maskinvare, blant annet arbeidsstasjoner eller parallelle maskiner.

Media Net er et ekstra protokoll-lag som kan legges oppå nettverkslag som f.eks. TCP, UDP, X.25 eller RS232. Media Objects er programvare for produksjon av interaktive tjenester.

3.2.3 Silicon Graphics: WebFORCE Media Server

SGI tilbyr en videotjener kalt WebFORCE Media Server, som er ment for bruk til blant annet leveranse av video på intranet (bedriftsinterne internett). Den kan levere videostrømmer på mellom 1.5 og 8 Mbit/s til klienter, via UDP. Det som er spesielt med WebFORCE er at tjeneren kan overvåkes og konfigureres via et WWW-grensesnitt. Det er i dag forholdsvis mange begrensende faktorer for WebFORCE, men det kommer en ny versjon (2.0) av WebFORCE i mars d.å. Blant annet har man ikke mulighet for tidskoding i filmene som skal vises. Det er heller ingen støtte for tertiære lagringsmedier, men dette skal komme med i senere versjoner. I versjon 2.0 kommer disse nye egenskapene [9].

- RAID-støtte
- transport via AAL5

[8] Relasjonsdatabasen som hovedproduktet til Oracle, den nåværende versjonen kalles stort sett bare Oracle7 (snart kommer Oracle8).

[9] Dette ble presentert av SGI-representanter på et møte hos Telenor i Trondheim 1996-10-25.

- støtte for Sun SPARC og Apple Mac-maskiner
- tidskoding
- støtte for H.261 (tidligere bare MPEG-1 og MPEG2)
- automatisk fordeling av nettverksbelastning
- åpne API-er for klientapplikasjoner
- lasting av innhold fra bånd og nettverk inn i diskbasen.

WebFORCE baserer seg altså på lagring på harddisk, og benytter SGIs XFS (Silicon Graphics Corporation 1997), et 64 bits skalerbart filsystem med støtte for blant annet garantert sanntids-overføring av data.

3.2.4 Sun MediaCenter Server

Sun leverer en løsning kalt MediaCenter Server som er basert på en kraftig maskin med UltraSPARC CPUer (Sun Microsystems 1996). Følgende programvare leveres sammen med tjeneren:

- En modifisert operativsystemkjerne (SunOS/Solaris, selvsagt).
- Spesielle nettverksdrivere, satt av til overføring av multimedia.
- Et spesielt filsystem kalt MFS (Media File System), som støtter levering av isokrone bit-strømmer.
- Media Stream Manager (MSM) med et RPC-basert API som gjør brukere i stand til å kontrollere strømmer spilt fra tjeneren.
- Content Manager (CM), som også har et RPC-basert API. CM gjør brukerne i stand til å laste data inn på tjeneren.

3.2.5 The Berkeley Distributed Video-on-Demand System

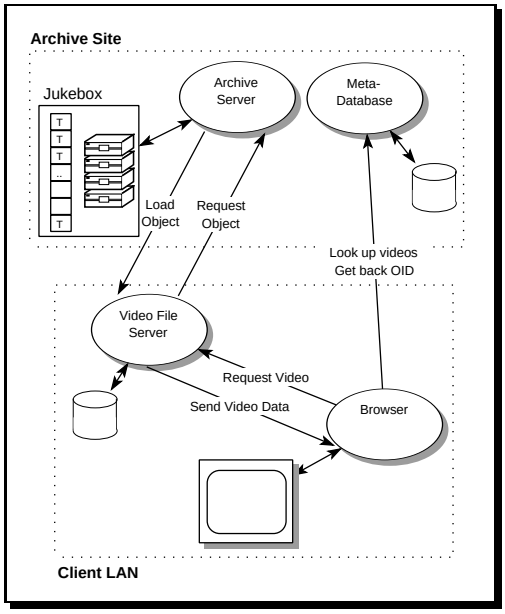
Berkeley Distributed Video-on-Demand System, forkortet til VODS er vel det systemet som ligger nærmest opp til IDIs eget VideoSTAR/Elvira-system. I motsetning til de kommersielle videotjenerene så har man også fokusert på at systemet skal kunne brukes som et videoarkivbibliotek.

De viktigste elementene i systemet er:

- OID – globale objektidentifikatorer som identifiserer ethvert lagret objekt (videofilm) i systemet.
- Archive Servers (AS) – Dette er en tjener eid av en eller flere produsenter av kontinuerlige multimedia. Ved å oppgi en OID (globale objektidentifikator) kan AS levere en film til en Video File Server (VSF) over LAN eller WAN. En AS kontrollerer en eller flere båndjukebokser.
- Meta Database – En metadatabase for strukturert informasjon om hva som befinner seg på AS. De bruker Postgres, som er en gratis RDBMS utviklet på Berkeley (Fournier 1997).
- Video File Server – En filserver som kan betjene flere videospillere på en gang, over et lokalnett. VFS-ene har harddisk-reservoarer der filmer blir bufret.
- Browser – Programmer som brukerne kjører for å få aksess til de lagrede dataene.

Prinsippet er at produsenter av kontinuerlige multimedia legger ut produktene sine på sin egen Archive Server og registrerer innslagene i metadatabasen.

Administratorene for et LAN som skal benytte seg av VODS (som kan ligge langt borte, og bare kan nås via et WAN), må sette opp en VFS som bufrer video før brukerne kan aksessere materialet.



Figur 3.4: Prosessdiagram for Berkeley VODS (Federighi og Rowe 1994)

Det er kun VFS-er som har overføringskapasitet til å betjene brukere real-time. Det kan derfor bli ganske lange ventetider før man får inn en film i en VFS dersom den ikke ligger der fra før av. Når filmen er blitt bufret i en VFS, kan den spilles av til en eller flere brukere som befinner seg på det LAN-et som VFS ligger på. VFS-er som befinner seg nær hverandre nettverkstopologisk, kan samarbeide om å bufre filmer. For å bufre effektivt, er VFS-ene avhengige av at brukerne spesifiserer hva de har tenkt å se på på forhånd.

På en måte ligner Berkeley VODS ganske mye på VideoSTAR (metadatabase) og Elvira (med ESM) når det gjelder høynivå konstruksjon. Den største prinsipielle forskjellen er den distribuerte tankegangen. Dette kan være en aktuell utvidelse i fremtiden, se avsnitt 9.3, side 110.

3.2.6 Konsekvenser

Det eksisterer allerede mange løsninger som er langt mer utviklet enn det man kan forvente at et Elvira med ESM vil bli. Meningen med Elvira/ESM er ikke å skape et system som skal konkurrere med kommersielle applikasjoner, men heller å være et fundament for videre forskning innen masselagring av multimedia.

3.3 Digital video og audio

Dette avsnittet tar for seg formater brukt i audio- og video-sammenheng. I dag har Elvira støtte for Motion JPEG og MPEG-1, så det er disse standardene som er mest interessante å diskutere her. En liten oppsummering er gitt i tabell 3.3.

3.3.1 Motion JPEG

JPEG står for Joint Photographic Experts Group. JPEG-standarden (ISO/IEC 10918) er en standard for komprimering av *stillbilder*. Den er ganske effektiv, og den tar mange parametre som styrer blant annet hvor kraftig bildet skal komprimeres.

Det finnes ingen standard for video basert på JPEG-stillbilder. Motion JPEG (M-JPEG) leveres av flere produsenter, men de forskjellige M-JPEG formatene har en tendens til å være inkompatible. (N.N. 1997).

Når bilder blir komprimert med JPEG, vil man ikke få full kvalitet tilbake når man dekomprimerer dem. Dette betegnes som ikke-tapsløs komprimering (non-lossless)[10].

M-JPEG har, til tross for manglende standardisering og forholdsvis lite komprimering i forhold til MPEG, vært ganske populært i sammenheng med videoredigering (Hjelsvold 1995, Laursen et al. 1994). Fordelen med M-JPEG er nemlig at hver videoramme er lagret uavhengig av hverandre, og man trenger ikke å generere bilder ved spoling, samt at man ikke må re-kode forholdsvis mange rammer når man setter inn nye rammer, slik som i MPEG.

Båndbredde på M-JPEG-strømmer er gjerne mellom 2 og 8 Mbit/s, avhengig av hva slags kvalitet som ønskes.

3.3.2 MPEG

MPEG står for Moving Picture Experts Group (Hjelsvold 1995, Dybvik 1996)[11]. Denne komiteen har laget en familie standarder for digital koding av video og audio, som er godkjent av IEC og ISO som internasjonale standarder. Under følger en beskrivelse av de to mest kjente MPEG-standardene[12].

MPEG-1 (ISO/IEC-11172) er en standard for digital lagring og lesing av bevegelige bilder og lyd på lagringsmedier. Kort fortalt har man komprimering både i tid og rom. Prinsippet er at det er kun endringer fra bilde til bilde som lagres. I prinsippet kunne man man startet med ett komplett bilde, og deretter kun følge på med endringer, men dette gjør at spoling blir nærmest umulig. Derfor settes det inn såkalte I-rammer (intracoded frames) som inneholder et komplett bilde og er kodet på en JPEG-lignende måte.

Video komprimert med MPEG-1 er ment å ha en VHS-lignende kvalitet og en overføringsrate på omkring 1.5 Mbit/s. Det er mulighet for to lydspor (stereolyd) med båndbredde på rundt 250 kbit/s. Når man har multiplekset MPEG-1 video og lyd, kalles det en MPEG-1 systemstrøm (Dybvik 1996). Den totale båndbredde for slike strømmer blir altså på rundt 1.8 Mbit/s. Bildekvaliteten har en øvre grense på 352x240 pixler, og en øvre rammefrekvens på 30 Hz (non-interlaced). MPEG-1 er først og fremst en lavkvalitets-standard optimalisert for CD-ROM.

MPEG-2 (ISO/IEC-13818) er en standard for komprimering av video rettet mot digital kringkasting. Den er ment for høyere båndbredder (opptil 40 Mbit/s), har støtte for flere rammeformater og den takler interlaced video, i motsetning til MPEG-1 (Lo 1997). Et eksempel på kvalitet kan være MPEG-2 for HDTV-applikasjoner, som har en grense på 1920x1080 pixler og rammefrekvens på 30Hz. Dette var tidligere den planlagte MPEG-3 standarden, som ble innlemmet i MPEG-2 fordi MPEG-2-formatet viste seg å fungere for HDTV.

I likhet med MPEG-1 har man også standarder for overføring av lyd. MPEG-2 systemstrømmer kan inneholde 5 audiostrømmer, i tillegg til videostrømmen. 5 audiostrømmer muliggjør overføring av surround-lyd.

3.3.3 Andre formater

Det finnes også endel andre formater for bevegelige bilder (Hjelsvold 1995).

- H.261

[10] JPEG-standarden spesifiserer også en tapsløs algoritme, men denne er lite brukt.
[11] Det er en vanlig feiloppfatning at M-en står for Motion
[12] Det finnes også flere (MPEG-4 og MPEG-7) som tar for seg andre aspekter av multimedia (Savatier og Fogg 1997, Lo 1997)

Nivå	Kvalitet	Bitrate	Typisk bruk
«Low»	352x240x30	4 Mbit/s	Forbruker-TV
«Main»	720x480x30	15Mbit/s	TV-produksjon
«High 1400»	1440x1152x30	60Mbit/s	Forbruker-TV
«High»	1920x1080x30	80Mbit/s	HDTV-produksjon

Tabell 3.2: Kvalitetsnivåer i MPEG-2

- QuickTime
- DVI
- CD-I

De tre siste formatene er proprietære standarder eid av enkeltselskaper (respektivt Apple, Intel og Philips). H.261 ligner litt på MPEG-1 i oppbygning, men har dårligere støtte for interaktive applikasjoner og er først og fremst ment for bruk i forbindelse videokonferanser[13], bildetelefoner og lignende (Hjelsvold 1995).

3.3.4 Litt om komprimering

Båndprodusenter (som Tandberg Data) skryter av at de har komprimering som gjennomsnittlig skal klare å komprimere ned datamengden til omtrent halvparten av den opprinnelige, selvsagt uten tap av informasjon. Når man skal lagre audiovisuelle data, er dette noe man må ta med en stor klype salt, fordi vanlige, generelle komprimeringsalgoritmer som f.eks. Huffman (Cormen et al. 1990) har ikke særlig effekt for videodata. Pikselverdier varierer over store intervaller (Lo 1997), og det er derfor vanskelig å finne mønstre gjentar seg (og som dermed kan slåes sammen). Uansett så har gjerne komprimeringsalgoritmene for lyd og video benyttet seg av disse algoritmene allerede.

De filene man har lagret på harddisk som MPEG-1 eller M-JPEG kan derfor med stor sannsynlighet forutsettes å bruke like mange bytes på bånd. Det er faktisk mulig at volumet kan øke noe, på grunn av tilleggsinformasjon som må lagres ved hver komprimering.

3.3.5 Oppsummering

I tabell 3.3 finner man en kort oppsummering av formatene diskutert i dette avsnittet.

Type	Overføringsrate	Kvalitet	Enkelt å redigere	Typisk bruk
M-JPEG	2-8 Mbit/s	Middels	Ja	Redigering
MPEG-1	1.8 Mbit/s	Lav	Forholdsvis	CD-ROM
MPEG-2	4-15 Mbit/s	Høy	Forholdsvis	Kringkasting (ikke HDTV)
H.261	64kbit/s	Lav	Nei	Konferanser

Tabell 3.3: Oppsummering av digitale videoformater

3.3.6 Konsekvenser

Siden det finnes flere standarder, er det best om ESM gjør seg så uavhengig av disse som råd er. Følgende bør gjøres:

[13] f.eks. over Multicast backbone, det såkalte MBONE, som også tilbyr lyd og bilder fra NASAs romferder. Alle med en høykapasitetstilkobling til Internett kan bruke MBONE.

- Mesteparten av systemet bør behandle dataene som rene rådata, der man ikke tar hensyn til struktur i det hele tatt.
- Der det er uunngåelig å la være å ta hensyn til dataformatet, bør det introduseres en modulmekanisme. Denne mekanismen gjør at dersom man skal lagre et nytt format, så lager man kun en ny modul for denne.
- Grensesnittet mod båndstasjonen bør ha mulighet til å slå av komprimeringsfunksjonen.

3.4 Fjernsynsprodusenter og digital video

Fjernsynsprodusenter er i ferd med å konvertere fra analogt til digitalt produksjons- og lagringsutstyr. Det er derfor interessant å studere hva slags erfaringer og hva slags forventninger slike produsenter har om f.eks. videoarkiver og videotjenere.

3.4.1 Norsk Rikskringkasting

Norsk Rikskringkasting (NRK) har i dag et stort arkiv, der materialet må transporteres ut fysisk. Det har allerede vært et forprosjekt i NRK når det gjelder et nytt arkiv, og man legger nå planer for å bruke flere millioner kroner på et nytt. Faglærer og undertegnede besøkte NRK 1996-12-13, og møtet med de syv ansatte er referert i tillegg B. Kort oppsummert resulterte møtet i følgende informasjon:

- NRK har et sentralarkiv for fjernsyn, et for radio og i tillegg har hvert distriktskontor egne arkiver.
- Det vil være aktuelt for NRK å tilby salg av materiale fra disse arkivene i fremtiden. NRK har produsert materiale i lang tid, og dette har markedsverdi, dersom det kan gjøres tilgjengelig på en enkel måte.
- Det ligger omtrent 90 000 timer videomateriale i fjernsynsarkivet. Dette er lagret på forskjellige typer media (både film og magnetbånd) i forskjellige formater.
- Per 1996-01-01 er det rundt 150 000 dokumenter registrert i en dokumentdatabase kalt SIFT.
- Det legges inn omkring 1000 nye dokumenter i året. Dette tallet er forventet å øke i fremtidien, blant annet på grunn av introduksjonen av NRK2.
- Det er viktig for NRK å ha muligheten til å raskt finne ut hvem som sitter med rettighetene til et bestemt materiale.
- Det er kun rundt 20 mennesker som til daglig er direkte brukere av arkivet.
- I 2/3 av forespørslene mot arkivet ber brukerne om aktuelt stoff.
- Det hentes ut mange flere bånd enn det som trengs, fordi man ikke er sikker på hvilket bånd materialet man leter etter befinner seg på.
- Nyheter og sport registreres mye mer nøyaktig enn andre former for programmer.
- En eventuell interaktiv digital videoarkivtjener skal kun lagre materiale i såkalt påsynskvalitet (f.eks. 2 Mbit/s). Produksjonskvalitet krever langt høyere lagringsmengder. Overføringsraten på produksjonsvideo er for tiden på hele 270 Mbit/s (Digital Beta).

3.4.2 Konsekvenser

Elvira (med ESM) kan benyttes som et rammeverk for diskusjoner omkring anskaffelse av en videoarkivtjener i NRK. Det er ingen tvil om at digital behandling av video er på full fart inn i de fleste fjernsynsselskaper, og at det er snakk om betydelige investeringer når disse selskapene skal fornye utstyret og produksjonsprosessene sine.

3.5 Forskning på tertiære lagringsmedier

.

Det foregår forskning på bruk av tertiære medier til mange formål. Her presenteres fire prosjekter som har en viss relevans for arbeidet med ESM.

3.5.1 Optimalisering av tilfeldige aksesser på bånd

Det er utført detaljerte studier av en en DLT båndstasjon (Silberschatz og Hillyer 1996*a*, Silberschatz og Hillyer 1996*b*). Dette ble gjort for å for å verifisere en egenutviklet algoritme for effektiv aksessering av serpentinbånd.

Modellen benytter følgende oppdeling av et bånd: Et bånd har flere spor, hvert spor har flere seksjoner, og hver seksjon har flere segmenter (blokker av fast størrelse). DLT har to modi for «lesing», nemlig søking (rask lesing av data, ikke særlig sikkert) og virkelig lesing (tregere lesing av data, men sikkert)[14].

Silberschatz og Hillyer (1996*a*) prøvde flere algoritmer for å oppnå en effektiv ressursfordeling når man hadde mange uavhengige aksesser til data på ett bånd. Disse ble prøvd:

- READ – lese hele båndet sekvensielt og laste ned de dataområdene man leter etter.
- FIFO (first in, first out)– behandle forespørslene i den rekkefølgen de ankommer.
- SORT – sortere forespørslene etter logisk blokknummer. O($nlgn$) i kjøretid, som er vanlig for sorteringsalgoritmer. n er antall forespørsler.
- SLTF (shortest locate time first) – En «greedy»-algoritme med kjøretid O(n^2).
- WEAVE – en tilnærming til SLTF, som gjør at den får O(n) i kjøretid.
- SCAN (elevator-algoritmen) – stasjonen leser båndet att og fram, og leser inn data når den kommer til en blokk som er forespurt.
- LOSS – en såkalt «greedy»-algoritme for løsing av det såkalte «traveling salesman»-problemet. O(nlg^2n) i kjøretid.

For å kunne måle effekten av algoritmene de hadde laget for ressursfordeling, måtte de modellere lokaliseringstiden på en DLT-stasjonen de brukte, og dette var hva de kom frem til:

> [Når man skal aksessere en blokk], flytt leshodet til sporet blokken befinner seg på, søk forover eller bakover helt til man treffer en seksjonsgrense som er to nøkkelposisjoner foran blokken som skal leses. Les så forover til man når denne blokken.

3.5.2 Striping av data på bånd – RAIL

Data kan stripes over et hvilket som helst medium, ikke bare harddisker, slik vi har sett i avsnitt 3.1.2.3. Prinsippene skissert i avsnitt 1.2, side 2 gjelder også for bånd, og det er gjort endel forskning på dette området.

På samme måte som ved RAID, ønsker man å stripe dataene over flere bånd (eller båndbiblioteker), slik at man får høyere I/O båndbredde og kortere responstid (Drapeau og Katz 1993). Dersom man i tillegg ønsker å lagre paritetsblokker, som gjør rekonstruksjon ved båndfeil mulig, har man fått et RAIL (Redundant Array of Independent Libraries) (Ford et al. 1996).

[14] Dette gjelder også for Tandberg Datas 2.5GB QIC-stasjon, mens MLR-1 har samme hastighet under både lesing og søking.

Forskere på IBM Almaden Research Center har implementert et programvarebasert RAIL i C++ (Ford et al. 1996). I en prissammeligning mellom et stort enkeltbibliotek på 188GB og 9 enkeltbåndstasjoner satt sammen som en RAIL med en aggregert kapasitet på 188 GB, var prisforskjellen per megabyte liten (enkelenheten var 10 cent/MB billigere), mens forskjellen i prisen for overføringsrate var halvert for RAIL (2.17 dollar/MB/s mot 1.0 dollar/MB/s).

3.5.3 Store lagersystemer

NASA har et prosjekt kalt *ECS* (EOSDIS Core System) som skal gjøre det mulig å hente ut data fra et enormt datalager som inneholder over en peta byte (10^{15} bytes). Disse dataene blir generert av sonder og instrumenter som observerer jorden, i et prosjekt kalt EOSDIS (Earth Observing System Data and Information System). NASA har satset på å bygge et system av hyllevarekomponenter[15] og unngår høye kostnader som spesiallagede komponenter vil innebære (Caulk 1995). Systemet er fortsatt under utvikling, men høynivå-designet viser at man vil benytte seg av store båndbiblioteker, også i arbeidslagere (ikke bare i arkivet). Det er mulig at disse vil bli stripet, som diskutert i avsnitt 3.5.2.

3.5.4 Hjemmesnekrede RAIDs

På Berkeley har man en forskningsgruppe som studerer bruk av disk som tertiærlager, istedenfor å bruke near-line båndbiblioteker (Asami et al. 1995). Man har blant annet implementert et RAID bestående av 112 disker, betjent av 2 PC-er og koplet sammen med diverse SCSI og PCI-busser.

Det blir påstått at på grunn av kostnadsutviklingen for harddisker og båndstasjoner, vil hjemmesnekrede RAIDs bli billigere per megabyte enn båndbiblioteker.

Forfatterne har imidlertid tatt utgangspunkt i at det skal være en ganske høy I/U-kapasitet på slike systemer, og da trenger man dyre biblioteker med mange stasjoner. Et annet moment er at harddisker blir masseprodusert i langt høyere grad enn båndstasjoner. Det kan derfor være snakk om en *kunstig* prisfordel for harddisker, skapt av markedet. Dersom det ble skapt et stort marked for f.eks. båndbiblioteker, er det mulig at prisene ville blitt presset nedover, slik at den virkelige kostnaden mellom harddisker og båndbiblioteker kom til syne.

Et båndbibliotek har også den fordelen at båndene (mediet) ikke er utsatt for mekanisk slitasje når de er off-line. Harddisker er derimot i aktivitet hele tiden. Det er viktig å se på brukstimer når man beregner forventet tid til feil. Når en robotarm i et bibliotek blir ødelagt, kan hele tjeneren stoppe opp, men selve dataene ligger trygt. Når en harddisk krasjer, er det derimot fare for at data går tapt.

3.5.5 Konsekvenser

Forskningen til Silberschatz og Hillyer (1996*a*) kan føre til at henvendelser om video fra samme bånd kan ordnes på en effektiv måte, dersom man finner en god modell for hvordan båndstasjonen som skal benyttes oppfører seg. Store systemer for aktiv bruk av enorme mengder data er under utvikling. Dette kan føre til at det blir skapt et større marked for båndteknologi, som igjen kan føre til at spådommen om at harddisker vil overta for båndlagring blir gjort til skamme. Striping av data over flere bånd er en komplisert prosess som ikke er aktuell for denne hovedoppgaven, men som kan vurderes i et senere system.

[15] ofte kalt COTS products på engelsk (commercial, off-the-shelf products).

3.6 Oppsummering

Her følger en punktvis oppsummering av de viktigste faktaene fra dette kapittelet:

- ESM bør ikke gjøre seg avhengig av et spesifikt tertiært lagringsmedium, siden det er en rivende utvikling av nye produkter innen dette området av lagringsteknologien.
- Mange forskjellige formater for komprimert lagring av mediadata gjør at ESM bør være modulær. Man bør abstrahere seg bort fra formater der det er mulig å gjøre det, slik at videotjeneren blir enkel å oppgradere senere.
- NRK har vist sin interesse for samarbeid med IDI om forskning på video(arkiv)tjenere.
- Det skjer fortsatt en rivende utvikling på området masselagring.
- Elvira er ikke ment å skulle konkurrere mot kommersielle implementasjoner av videotjenere. Elvira er ment å fungere som et kunnskapsoppbyggende prosjekt.

Kapittel 4

Modell av et digitalt videoarkivsystem

I dette kapittelet presenteres en modell for oppførselen til et digitalt videoarkiv på forskjellige nivåer, med fokus på lagring på bånd og båndstasjoner, siden det er dette som er hovedtemaet for denne oppgaven.

Hensikt

En modell er en forenkling og abstraksjon av et komplekst system som tjener følgende formål:

- forenkling
- kommunikasjon
- testgrunnlag
- konstruksjonsgrunnlag

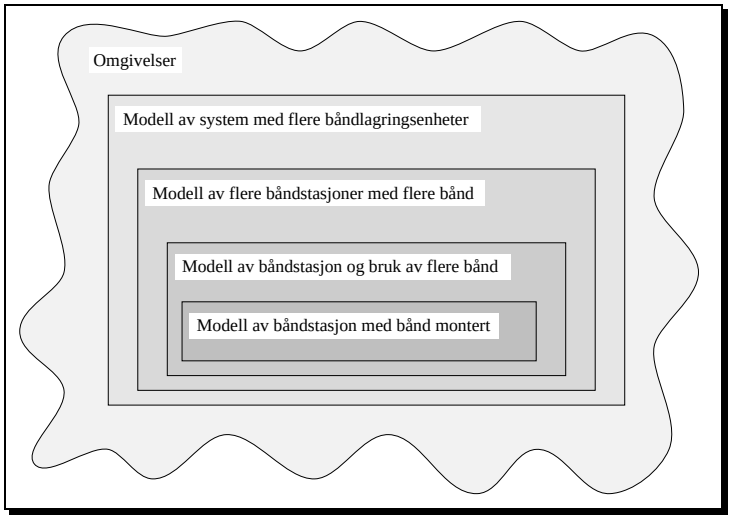
Modellen er en *forenkling* av systemet fordi den kun tar med de viktigste parameterne og strukturene i systemet. Denne forenklingen gjør at det blir lettere å *kommunisere* med andre om problemstillinger knyttet til digitale arkivsystem. Siden forenklingen inneholder de parameterne som påvirker systemets egenskaper i størst grad, er modellen et *grunnlag for testing*.

Oppbygging av modell

Modellene blir presentert i en fra-bunnen-og-oppover-rekkefølge, eller innenfra og ut, slik som vist i figur 4.1. Først behandles et enkelt system der man har et bånd montert i en stasjon. Deretter utvides modellen med mulighet for lagring på flere bånd. Så introduseres muligheten for å ha flere båndstasjoner som arbeider i parallell mot samme arkiv. Til slutt beskrives et videotjenersystem bygget på de forannevnte mulige former for båndlagring.

Navngiving

For deterministiske variable brukes navn med kun små bokstaver, mens stokastiske variabelnavn skrives med store bokstaver. Deterministiske variable er variable som kan fastslåes med (svært) liten feilmargin fra forsøk til forsøk. Stokastiske variable er variable som kan anta en tilfeldig verdier for et gitt forsøk, men som følger en statistisk distribusjon i det lange løp. Variable har følgende indekser:



Figur 4.1: Oversikt over modellen

- *sd* – stasjon (engelsk: *s*ingle *d*evice).
- *lib* – båndarkiv (engelsk: *lib*rary).
- *lmd* – arkiv med flere stasjoner (engelsk: *l*ibrary with *m*ultiple *d*evices)
- *sys* – digitalt arkivsystem med lagring på bånd (engelsk: *sys*tem)
- *vp* – *v*ideo*p*umper, som er ansvarlige for å overføre data fra harddisker til videospillere via nettet.

Antagelser

Dersom målinger/beregninger skal utføres, bør man operere med verdier som går på ukomprimert lagring, siden båndstasjonens komprimeringsmekanisme ikke er særlig egnet til komprimering av kontinuerlige mediadata, som vist i avsnitt 3.3.4, side 24. For å unngå usikkerheten relatert til komprimeringen, er det best å slå den av under målinger.

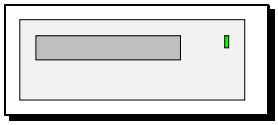
Når det gjelder overføringsrater for de tre første modellene, så går man ut i fra at vertsmaskinen og systemet er i stand til å motta data så raskt at disse ikke påvirker ytelsen.

Begrensninger i beregningene

Beregningene som er foretatt er omtrentelige. Det er ikke meningen å utføre en komplett matematisk/stokastisk analyse, siden dette ligger på siden av oppgaven. Derimot skal alle vesentlige parametre som beskriver systemet være behandlet, slik at en slik eventuell fremtidig analyse med hensyn på ytelse, kapasitet, robusthet og kostnader blir gjort enklere. Kostnadsparametre er forøvrig utelatt fra diskusjonen, fordi de er diskutert til en viss grad i forstudiet.

4.1 Båndstasjon med ett bånd montert

På det laveste nivået har man en enkelt båndstasjon med ett bånd montert i stasjonen, slik som vist i figur 4.2.



Figur 4.2: Båndstasjon med ett montert bånd

De viktigste parametrene for et slikt system er beskrevet i boksen under. De fleste variablene trenger ikke noen videre forklaring enn det som står der. Overføringsraten (T_{sd}), aksesstiden (A_{sd}) og feilratene bør imidlertid diskuteres ytterligere, fordi det er disse som til syvende og sist har avgjørende betydning for hvordan systemet vil oppføre seg.

Båndlengde (l_{sd}, length) er kun avhengig av hvilken type bånd som blir brukt.

Kapasitet (c_{sd}, capacity) som f.eks. kan måles i antall bytes som maksimalt kan lagres på et bånd av lengde l_{sd}. Den vil være konstant så lenge ikke for mange blokker er ødelagt, fordi båndstasjoner som regel har avsatt plass til reserveblokker med jevne mellomrom på båndet. Båndets kapasitet er selvsagt også avhengig av typen båndstasjon som benyttes.

Blokkadresse (B_{sd}, block address) er blokkadressen til dataområdet som skal hentes.

Posisjon på båndet (P_{sd}, position) sier hvilken fysisk posisjon lesehodet har i det båndspilleren får forespørselen. Denne variabelen er avhengig av blant annet B_{sd} og D_{sd} for forrige forespørsel.

Spolingshastighet (v_{sd}, velocity) er en parameter som vil være rimelig konstant fra stasjonstype til stasjonstype. Spolingshastigheten kan være avhengig av l_{sd} og P_{sd} dersom man ikke har konstant spolingshastighet (jf. hastighetsøkningen når man spoler tilbake en vanlig VHS-kassett i en videospiller). QIC har konstant spolingshastighet, på grunn av mekanikken som blir brukt i båndkassetten og i stasjonen.

Datamengde (D_{sd}, data size) angir hvor mye skal leses av båndstasjonen. Dette er en stokastisk variabel som er avhengig av hvor mye materiale brukeren ønsker å hente ut.

Overføringsrate (T_{sd}, transfer rate) som forteller hvor mange bytes som kan leveres fra båndstasjonen hvert sekund ved kontinuerlig lesing.

Aksesstid (A_{sd}, access time) er tiden det tar å posisjonere lesehodet til B_{sd}, slik at lesing kan begynne.

Forventet tid til blokkfeil ($MTTBE_{sd}$, mean time to block error) sier noe om hvor ofte båndstasjonen oppdager at en blokk er ødelagt og må repareres.

Forventet blokkrepareringstid($MTTBR_{sd}$, mean time to block repair) angir hvor lang tid båndstasjonen bruker når den må stoppe opp ved lesefeil.

Forventet tid til stasjonsfeil ($MTTF_{sd}$, mean time to failure) sier noe om hvor ofte en stasjon vil settes ut av drift på grunn av feil.

Forventet reparasjonstid for stasjon ($MTTR_{sd}$, mean time to repair), sier noe om hvor lang tid tar det å reparere stasjonen når den først feiler.

4.1.1 Overføringsrate

Overføringsraten (T_{sd}) i en typisk serpentinbåndstasjon ligger på mellom 0.5 og 3.0 MB/s (Bratbergsengen 1996, Saha 1996). Arbeid utført av Sætre og Sandstå (1997) viser at man kan få en jevn strøm av videodata på 284kB/s fra en Tandberg TDC 4220. En MLR-1 vil ha en overføringsrate på omtrent 1.5 MB/s, men dette er ennå ikke målt av oss.

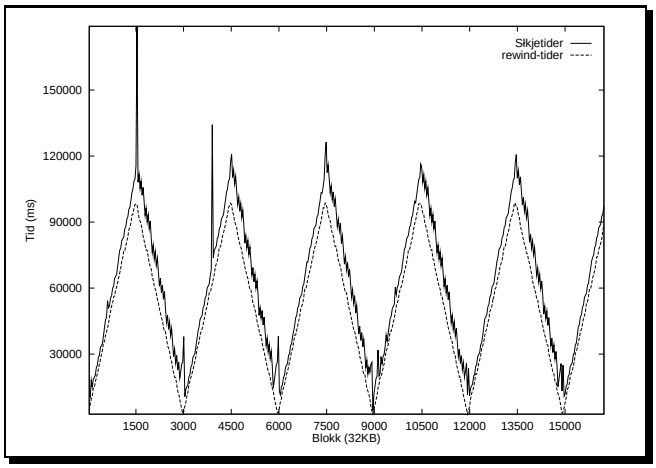
Spesifikasjoner på overføringsrater er gjerne noe optimistiske fra produsentenes side. Ofte opererer man med effektiv overføringsrate for komprimert materiale, som gjerne er dobbelt så høy som den

man får dersom man skal lagre kontinuerlige mediadata som allerede er komprimert, f.eks. MPEG-1 systemstrømmer.

Overføringsraten er også avhengig av $MTTBE_{sd}$ og $MTTBR_{sd}$ og hvilken type båndstasjon man har. Såkalte «bad blocks» fører til at båndstasjonen må reposisjonere seg. Hvor lang tid «reparasjonen» tar er avhengig av stasjonstypen, blant annet hva slags feilkorrigering som benyttes. Dersom slik reposisjonering vil skje ofte, vil den effektive overføringsraten selvsagt bli mindre.

4.1.2 Aksesstid

Databånd er sekvensielle media, noe som medfører at ulike adresser gir ulike aksesstider. På bånd som er strengt sekvensielle, slik som 8-mm og DAT, har man alltid at $P_{sd}(i) < P_{sd}(j) \Rightarrow A_{sd}(i) < A_{sd}(j)$ når båndet er tilbakespolt og man får forespørsel i og j. Dette gjelder ikke for serpentinbånd, på grunn av lagringmønsteret, slik som vist i 3.1.1.2, side 15. Man vil heller få et sagtannmønster som vist i figur 4.3. Aksesstiden vil altså være avhengig av P_{sd}, v_{sd} og algoritmen som båndstasjonen bruker for å lokalisere dataene.



Figur 4.3: Aksesstider for Tandberg TCD 4220 som funksjon av posisjon, B_{sd} (Sætre og Sandstå 1997)

Algoritmer som blir brukt for å lokalisering av blokker er regnet som forretningshemmeligheter, og dermed ikke er offentlig tilgjengelige. En modell for oppførselen til en båndstasjon er dermed en kvalifisert gjetning basert på antagelser og målinger på spesifikke stasjoner, slik som Silberschatz og Hillyer (1996*a*) har gjort. Disse vil ha den svakheten at de ikke kan forutsi hva som vil skje ned til minste detalj, siden det finnes et utall algoritmer produsenten har lagt inn i båndstasjonen for f.eks. feilhåndtering.

4.1.3 Feilrater

Hjørnestenen i en modell for feilrater er den såkalte tid-til-feil-distribusjonen, $f(t)$ (Dougherty 1990). Den stokastiske variabelen T angir tid til første feil. Et systems pålitelighet blir da karakterisert slik:

$$R(t) = P(T > t) = 1 - F(t) = \int_t^\infty f(x)dx$$

Dette sier at pålitelighet ved et tidspunkt t er sannsynligheten for at ingenting har gått galt i tiden før t.

Dougherty (1990) viser da at MTTF (mean time to failure), ($E[T]$, forventet tid til første feil) da blir:

$$MTTF = E[T] = \int_0^\infty R(t)dt$$

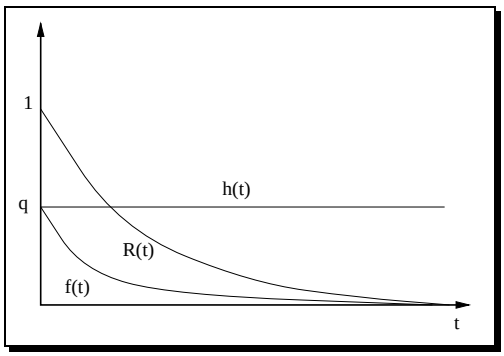
Hazard-funksjonen, $h(t)$ er gitt som $h(t) = \frac{R(t)}{f(t)}$ og sier noe om hvor stor sannsynlighet det skjer en ulykke akkurat ved tid t.

Dersom man teller antall feil over tid med en stokastisk variabel X, så vil man gjerne finne at X har en poisson-distribusjon, med parameter $\lambda = qt$, der q er *feilraten*. Teorem 4.12 i (Dougherty 1990) sier da at dersom Y er en stokastisk variabel som måler tiden mellom to feil, så har den eksponensiell distribusjon med parameter q, altså $f(y) = qe^{-qy}$. Det er svært vanlig at forekomster av feil over tid følger en poissonfordeling, og man kan derfor benytte seg av dette resultatet med stort hell.

For $f(t) = qe^{-qt}$ og $h(t) = q$ får vi:

$$R(t) = \frac{f(t)}{h(t)} = e^{-qt}$$

Et typisk system, eller en komponent i et system, vil ha en tid-til-feil-distribusjon som er eksponesielt avtagende, mens hazard-funksjonen er konstant (det er like stor sannsynlighet for en feil ved tidspunkt t_1 som ved tidspunkt t_2). Utviklingen av $f(t)$, $R(t)$ og $h(t)$ over tid er illustrert i figur 4.4. Merk at normalt vil q være flere størrelsesordner mindre enn vist her.



Figur 4.4: Feiltetthets-, pålitelighets- og hazard-funksjoner over tid

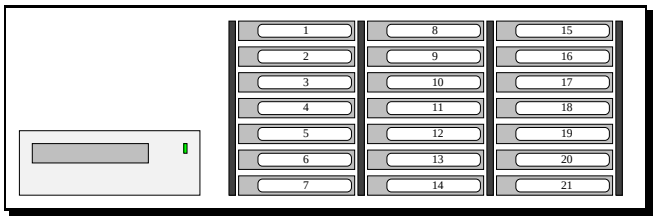
$MTTF_{sd}$ vil typisk være avhengig av tiden det tar før en elektrisk eller mekanisk feil oppstår, eller at hodet blir utslitt. $MTTBE_{sd}$ vil være avhengig av omgivelsene båndene blir lagret i og de magnetiske kvalitetene til båndet. $MTTBE_{sd}$ vil typisk være langt høyere enn $MTTF_{sd}$. Konsekvensen av feilene er imidlertid svært forskjellige. Dersom båndstasjonen feiler, er systemet helt nede. Dersom det oppstår en blokkfeil, kan denne (nesten alltid) rettes opp ved å benytte en ekstra informasjonen som båndstasjonen alltid skriver etter dataene, og som kalles ECC (error correcting code). Dersom feil oppstår, trenger stasjonene bare å flytte blokken til et reservert område. Dette fører til en liten forsinkelse, men det er ikke katastrofalt for systemet.

4.1.4 Oppsummering

ESM vil være avhengig av at tallene for overføringsrater og aksesstider er forholdsvis korrekte, og i hvertfall ikke underestimerte. Dersom det siste er er tilfellet, vil brukerne oppleve at systemet ofte ikke kan holde hva det lover. På den annen side er det viktig å ikke overestimere for kraftig, siden man da risikerer å få dårlig utnyttelse av systemet. Feilrater kan påvirke systemeffektiviteten, og ikke minst brukernes inntrykk av systemet.

4.2 Båndstasjon og flere bånd

Den enkle modellen ovenfor utvides med å tillate lagring på flere bånd, som kan settes inn i spilleren når man har behov for å få tilgang til dataene der. Man kan ha en manuell eller automatisk mekanisme som mater båndstasjonen med kassetter. Hva slags løsning som er valgt er uten betydning for modellen, annet enn at enkelte av parametrene antakeligvis vil anta svært forskjellige verdier.



Figur 4.5: Enkel båndstasjon og flere bånd

En båndstasjon er ikke særlig kostnadseffektiv (MB/NOK) før man kan fordele innkjøpskostnaden over en god del båndkassetter, som nevnt i avsnitt 3.1.1.

Tilbakespolingstiden (R_{lib}, rewind time) vil være avhengig hvilken posisjon på båndet siste lesning ble foretatt (P_{sd}). Det kan være mulig å forutsi denne tiden, siden man kan ha informasjon om hvor dataene befinner i en metakatalog i ESM. Selve tiden vil være avhengig av spolehastigheten til stasjonen (v_{sd}). Dermed får man følgende uttrykk: $R_{lib} = P_{sd} \cdot v_{sd}$.

Utkastingstid (e_{lib}, eject time) sier noe om hvor lang tid tar det å føre båndet ut av stasjonen, gitt at det er spolt tilbake. Dette vil kun være avhengig av stasjonstypen og variere lite fra gang til gang. Noen båndstasjoner vil bruke langt mer tid enn andre, fordi de har båndet montert opp på interne spoler i båndstasjonen (slik som DLT). Et QIC-bånd er derimot alltid i kassetten sin, og utkasting går raskt.

Lastingstid (l_{lib}, loading time) sier hvor lang tid tar det å laste båndet inn i stasjonen og gjøre det klart til bruk. Dette vil stort sett kun være avhengig av stasjonstypen og variere lite. En typisk lasting tar 10 sekunder på en QIC-stasjon.

Antall bånd (n_{lib}, number) i arkivet.

Kapasiteten (c_{lib}, capacity) kan finnes slik: $c_{lib} = n_{lib} \cdot c_{sd}$. Dersom man har redundans for å sikre data, vil tallet være noe lavere, men fortsatt deterministisk.

Posisjon i arkiv (P_{lib}, position) sier hvor båndet er lagret fysisk i arkivet.

Flyttetid (M_{lib}, moving time) Tiden det tar å flytte et bånd mellom arkiv og båndstasjon eller vice versa.

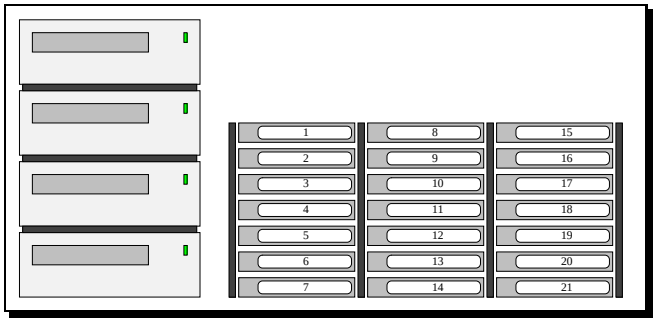
Veksletid (X_{lib}, exchange time) Tiden det tar å bytte et bånd i stasjonen med et annet.

Overføringsrate (T_{lib}, transfer rate) er antall bytes som kan overføres per sekund. Det er åpenbart at $T_{lib} = T_{sd}$.

Aksesstid (A_{lib}, accesss time) er det tar fra man har spesifisert hva man ønsker å lese til de første dataene ankommer.

Forventet tid til blokkfeil ($MTTBE_{lib}$), vil antakeligvis være noe mindre enn $MTTBE_{sd}$, siden båndene nå er lagret off-line mesteparten av tiden, og dermed ikke utsettes for mekanisk slitasje. Den kjemiske nedbrytingen av mediet er også lavere dersom båndene lagres i klimatempererte rom.

Forventet tid til flyttemekanismefeil ($MTTMF_{lib}$, mean time to mechanism failure) sier noe om hvor ofte en av flyttemekanismene vil feile.



Figur 4.6: Flere båndstasjoner og flere bånd

4.2.1 Veksletid

Tiden det tar å skifte ut et bånd med et annet, kan utledes av tiden det tar å:

1. spole det gamle båndet tilbake (dersom det er nødvendig).
2. kaste det gamle båndet ut av stasjonen.
3. føre det gamle båndet fra båndstasjonen og sette det inn på plassen sin i arkivet.
4. hente det nye båndet fra sin plass i arkivet og føre det til båndstasjonen.
5. sette det nye båndet inn i båndspilleren og la den montere det.

Man får dermed følgende uttrykk for veksletid: $X_{lib} = R_{lib} + e_{lib} + 2 \cdot M_{lib} + l_{lib}$ når dataene i to etterfølgende forespørsler ligger på forskjellige bånd og $X_{lib} = 0$ når de ligger på samme bånd. Ved båndveksling kan man redusere flyttetiden til det halve dersom byttemekanismen har mulighet til å holde to bånd av gangen.

4.2.2 Aksesstid

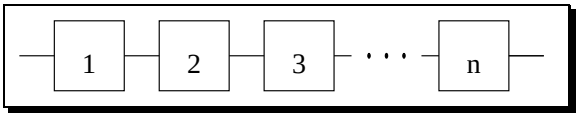
Aksesstiden, A_{lib}, blir nødvendigvis være endel høyere enn A_{sd} siden man normalt må bytte ut båndet som står i stasjonen med et annet et før man begynner søkingen. Aksesstiden kan uttrykkes som $A_{lib} = A_{sd} + X_{lib}$ dersom man må skifte bånd, og $A_{lib} = A_{sd}$ dersom båndskifte ikke er nødvendig, fordi man spør etter data fra et bånd som ble brukt i forrige aksess.

4.2.3 Oppsummering

Kostnadseffektiviteten vil bli bedre når man har mulighet for å ha et stort antall bånd som blir betjent av en båndstasjon. Aksesstiden vil øke forholdsvis dramatisk, slik at brukere risikerer å vente i lenger tid. Forventet tid til blokkfeil vil antageligvis gå ned dersom båndene lagres forsvarlig utenfor båndspilleren, siden lagringsmediet da vil unngå mekanisk slitasje mesteparten av tiden.

4.3 Multiple lagringsenheter

Modellen over utvides ved å introdusere en eller flere ekstra båndstasjoner, slik som vist i figur 4.6. Selv om man i prinsippet kan ha en ubegrenset mengde bånd i et arkiv, og kun en båndstasjon, vil dette konseptet nærmest være ubrukelig, fordi båndbredden mot arkivet vil være en flaskehals på grunn av at en enkelt båndstasjon maksimalt har noen få megabyte per sekund overføringsrate. Eksempelet i avsnitt C.3, side 126 illustrerer dette. Derfor må man introdusere ekstra båndstasjoner



Figur 4.7: Feilrater: Seriell konfigurasjon av komponenter

i systemet for å fjerne flaskehalsen på båndbreddesiden. Dette introduserer imidlertid endel nye prolemer som må studeres nærmere. I dette avsnittet antas det at alle stasjonene er av samme type.

Antall stasjoner (n_{lmd}, number) montert i arkivet.

Antall stasjoner ledig (F_{lmd}, free devices) på et bestemt tidspunkt.

Antall flyttemekanismer (m_{lmd}, moving devices) sier hvor mange bånd som kan flyttes uavhengig av hverandre på samme tid.

Veksletid (X_{lmd}, exchange time).

Overføringsrate (T_{lmd}, transfer rate).

Aksesstid (A_{lmd}, access time), som kan uttrykkes som $A_{lmd} = A_{sd} + X_{lmd}$ på lignende måte som i forrige avsnitt.

Forventet tid til bibliotekfeil ($MTTF_{lmd}$, mean time to failure) sier noe om hvor ofte systemet vil settes ut av drift på grunn av feil.

4.3.1 Veksletid

Dersom det er en ledig båndstasjon som kan flytte båndet ($F_{lmd} > 0$) vil man få følgende veksletid: $X_{lmd} = M_{lib} + l_{sd}$, ellers vil man få $X_{lmd} = X_{lib}$. Dette gjelder imidlertid bare om det finnes en flyttemekanisme for hver båndstasjon (dvs. $m_{lmd} = n_{lmd}$).

Dersom flere stasjoner deler en eller flere flyttemekanismer, dvs. $f_{lmd} < n_{lmd}$, og to eller flere bånd skal flyttes samtidig så vil man risikere å måtte vente på at en annen stasjon blir ferdig med å utføre sin utveksling.

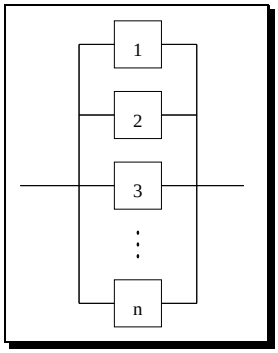
4.3.2 Overføringsrate

Hvor høy båndbredde man får i overføringen av en enkelt film er avhengig av hvordan dataene er organisert i arkivet. Dersom man har striping, kan man få opptil $T_{lmd}(i) = n_{lmd} \cdot T_{lib}(i)$ for en bestemt forespørsel i. Dersom dataene ikke er stripet, vil overføringsraten for den enkelte bruker være T_{lib}. Overføringsraten fra arkivet likvel være større, dersom man ser på det fra systemets side. Flere forespørsler kan nemlig betjenes samtidig.

4.3.3 Feilrater og organisering av lagring

Hvor ofte man får en stasjonsfeil som er så alvorlig at hele arkivet settes ut av spill, avhenger også av hva slags lagringsstruktur man har. Dersom man striper dataene over nøyaktig n_{lmd} bånd, vil systemet være nede så fort en av stasjonene feiler, noe som fører til uakseptabelt store sjanser for at systemet går ned. Dette er ekvivalent med et generelt system bestående av n_lmd komponenter koblet i serielt, slik som vist i figur 4.7.

Dersom man ikke striper i det hele tatt, har man derimot fått en garanti om at systemet alltid vil



Figur 4.8: Feilrater: Parallell konfigurasjon av komponenter

være i stand til å betjene forespørsler, så lenge ikke alle n_{lmd} stasjoner feiler på en gang! Dette er ekvivalent med et generelt system koblet i parallell, slik som vist i figur 4.8.

$R_i(t)$ brukes om pålitelighetsfunksjonen for hver enkeltkomponent og antar man at alle komponenter har samme pålitelighet, dvs. $R_i(t) = R(t)\ \forall i$. Da får man «reliabilityfunksjonen» for et serielt system med n komponenter, $R_{ser}(k,t)$, som følger:

$$R_{ser}(n,t) = P(T > t) = \prod_{i=1}^{n} R_i(t) = R(t)^n$$

Siden $R(t) < 1$ betyr dette at systemet er mindre til å stole på en den enkelte komponent som utgjør systemet, dvs. $R_{ser} < R(t), n \geq 2$. For et parallelt system får man:

$$R_{par}(n,t) = P(T > t) = 1 - \prod_{i=1}^{n}(1 - R_i(t)) = 1 - (1 - R(t))^n$$

Her kan man se at systemet kan stoles på mer enn den enkelte komponent, fordi $R_{par} > R(t), n \geq 2$. Disse utregningene er vist mer utførlig i avsnitt C.4, og er tatt fra Dougherty (1990). Dersom man har en eksponensialfordeling for tid-til-feil på $f(t) = qe^{-qt}$ får man $R(t) = e^{-qt}$ og en hazard-funksjon på $h(t) = q$. Dersom man antar at hazard-funksjonen konstant og lik for alle komponentene, får man følgende uttrykk for $MTTF_{ser}(n)$:

$$MTTF_{ser}(n) = E[T] \cdot \frac{1}{n} = \frac{MTTF}{n}$$

Og i det parallelle tilfellet:

$$MTTF_{par}(n) = E[T] \cdot \sum_{i=1}^{n} \frac{1}{i} = \sum_{i=1}^{n} \frac{MTTF}{i}$$

Dette kan tolkes slik:

- I et serielt system med n komponenter, kan man forvente at det oppstår feil i systemet etter en n-te del av forventet tid for feil i en enkeltkomponent.
- I et parallelt system med n komponenter kan man forvente at man får høyere forventet tid for feil i systemet enn for de enkelte komponenter. For store n blir forbedringen svært liten.

Man får også totalt stopp i systemet dersom mekanismen for utveksling av bånd feiler. Det kan derfor være nyttig å ha et system med flere flyttemekansimer der alle flyttemekanismene kan betjene alle stasjonene. Diskusjonen om feilrater for flyttemekanismer blir den samme som den vist over.

Forventet tid til bibliotekfeil ($MTTF_{lib}$) er altså avhengig av følgende:

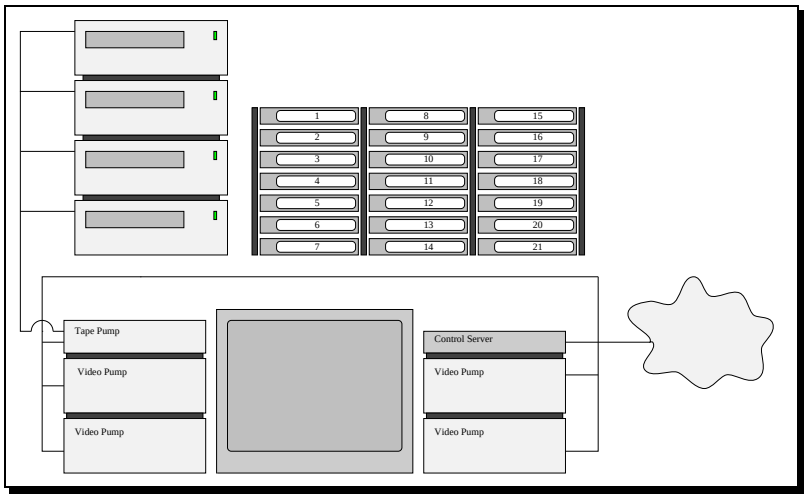
- Hva slags organisering man har på lagringen av data.
- Om det er redundans på alle komponentnivå i systemet.
- Feilratene til den enkelte stasjon ($MTTF_{sd}$) og antall stasjoner (n_{lmd}). Feilratene for den enkelte flyttemekanisme ($MTTMF_{lib}$) og antallet flyttemekanismer (m_{lmd}).

4.3.4 Oppsummering

Ved å bruke flere båndstasjoner som opererer mot det samme arkivet oppnår man en økt båndbredde. Hvordan brukerne opplever båndbreddeøkningen er avhengig av organiseringen av dataene (striping, ikke striping, etc). Feilrater er avhengig av hvordan man organiserer lagringsstrukturen, og hvor mange flyttemekanismer som eksisterer i systemet.

4.4 Videotjener med båndlagring

Dette avsnittet tar for seg en hel videotjener, der data lagres på digitale databånd. Det er realistisk å ha et system med en båndstasjon og mange bånd (avsnitt 4.2) eller et system med flere båndstasjoner og flere bånd (avsnitt 4.3). For enkelhets skyld bruker vi bare notasjonen for det siste tilfelle, og lar det første være et spesialsering av dette (dvs. vi setter antall båndstasjoner, $n_{lmd} = 1$). Mot slutten av dette avsnittet presenteres en overordnet beskrivelse av aktuelle kømodeller for en digital videoarkivtjener.



Figur 4.9: System med flere bånd og stasjoner

Her er det ikke bare karakteristikkene til båndarkivet som påvirker ytelsen, men også andre faktorer i selve systemet, slik som belastning på videopumpene). I figur 4.9 kan man se en digital videoarkivtjener bestående av et båndarkiv med fire stasjoner og kapasitet på 21 kassetter. Systemet har 4 arbeidsstasjoner med store harddisker. En videopumpe-prosess kjører på hver av dem. Gjennom disse videopumpene foregår *all* aksess mot harddiskene. I tillegg finnes en arbeidsstasjon som er vert for tjenerens kontroll-prosess og og enda en arbeidsstasjon avsatt til å betjene båndarkivet.

Når data skal overføres fra båndarkiv til harddisker, må det skje gjennom videopumpene, som da blir satt i «revers-modus». Alle maskinene er knyttet sammen med et høykapasitetsnettverk.

Overføringsrate mot videopumpe (T_{vp}, transfer rate) sier hvor hurtig tjenerens videopumper kan ta i mot data i reversmodus. Denne vil være avhengig av hvor mange videospillere som blir betjent.

Aksesstid mot videopumper (A_{vp}, access time) sier hvor lang tid systemet må vente før det får lov til å benytte videopumpene når data skal overføres fra bånd til harddisk. Dersom det alltid er reservert båndbredde for overføringer fra bånd til disk, eller dersom systemet ikke er fullt belastet, vil $A_{vp} \simeq 0s$. Dersom systemet ikke har satt av en slik båndbredde og systemet er tungt belastet, er det opp til kontroll-systemet å bestemme hvor stor A_{vp} blir.

Overføringsrate (T_{sys}, transfer rate) sier noe om hvor fort systemet kan flytte data fra bånd til harddisk.

Aksesstid (A_{sys}, accesss time) er tiden det tar fra systemet bestemmer seg for å overføre data fra bånd til disk, til de første dataene ankommer diskene.

Betjeningstid (S_{sys}, serving time) er tiden det tar å betjene en forespørsel, og er avhengig av aksesstid, overføringstid og mengde data som skal overføres. Dette kan uttrykkes som $S_{sys} = A_{sys} + T_{sys} \cdot D_{sd}$.

Ventetid i kø (W^q_{sys}, waiting time in queue) er tiden det tar fra en forespørsel er lagt inn i systemets kø og til den tas ut for å bli betjent.

Antall forespørsler i køen (L^q_{sys}, length of queue) sier hvor mange forespørsler som befinner seg i køen.

Ventetid før betjening er utført (W_{sys}, waiting time) er tiden det tar fra en bruker har bestilt en video fra bånd, til forespørselen er utført av systemet. Dette kan uttrykkes som $W_{sys} = W^q_{sys} + S_{sys}$.

Totalt antall forespørsler i systemet (L_{sys}) ved et bestemt tidspunkt. I et system med s båndstasjoner, kan det være opptil s forespørsler som er i ferd med å bli betjent, og L^q_{sys} forespørsler i køen. Dette kan uttrykkes som $L^q_{sys} + x$ der $x \in [0, s]$,

Ankomstrate for forespørsler (λ_{sys}) måles f.eks. i ankomster per sekund, per minutt eller per time. Ankomstraten er kun bestemt av omgivelsene til systemet (se neste avsnitt). Når vi ser på systemet i et stort tidsperspektiv, så forventer vi at denne er rimelig konstant, f.eks. fra dag til dag.

4.4.1 Aksesstid

Aksesstiden for systemet, A_{sys}, vil være avhengig av aksesstiden mot arkivet, A_{lmd} og aksesstiden mot videopumpene A_{vp}. Aksesstiden for en overføringsoperasjon som starter ved tidspunkt t kan uttrykkes slik:

$$A_{sys}(t) = A_{lmd}(t) + A_{vp}(t + A_{lmd}(t))$$

Det siste leddet sier at aksesstiden mot videopumpe må måles i det overføring fra båndet er klargjort, og dette skjer etter $t + A_{lmd}(t)$ tid. Dersom systemet har alltid er garantert direkte aksess til videopumper, eller belastningen er lav, vil uttrykket reduseres til $A_{sys}(t) = A_{lmd}(t)$, eller bare:

$$A_{sys} = A_{lmd}$$

Det vil si at aksesstiden for videomateriale i systemet er den samme som aksesstiden mot båndarkivet, men dette gjelder bare når vi alltid har en garantert tilgang til videopumpene for overføring av data fra arkiv til disk.

4.4.2 Overføringsrate

Overføringsraten for systemet, T_{sys} vil angi hvor fort systemet klarer å overføre data fra båndarkivet til harddisker via videopumpene. Man kan risikere at båndbredden på nettverket (T_{net}) som brukes til overføring mellom båndarkivet og videopumper kan være for liten til å betjene overføringen. Overføringsraten for systemet vil være lik minimum av overføringsrate for bånd, videopumper eller nettverk, og kan uttrykkes slik:

$$T_{sys} = min(T_{lmd}, T_{vp}, T_{net})$$

Vi har her antatt at overføringsraten er konstant for en overføring. Siden båndbredden for båndarkivet er såpass dyrebar (denne vil typisk utgjøre en flaskehals i en typisk videoarkivtjener), burde det allokeres fast båndbredde på videopumper og nettverk[1], og dermed kan uttrykket reduseres til:

$$T_{sys} = T_{lmd}$$

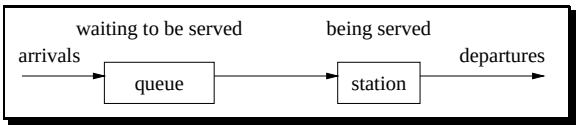
Det vil si at overføringsraten er kun bestemt av overføringsraten til båndarkivet så lenge vi alltid har reservert båndbredde på nett og videopumper som er lik eller større enn den båndbredden båndarkivet har.

4.4.3 Enkel kømodell

Det kan være svært interessant å lage kømodeller for en digital videoarkivtjener. Grunnen til dette er at det følgende parametere for systemet kan beregnes ved hjelp av statistiske metoder (Dougherty 1990, Taylor og Karlin 1994):

- Utnyttelsesgrad for tjeneren – viktig når man skal sette opp effektivitet mot kostnader.
- En distribusjon for antall brukere i systemet – kan gi en pekepinn om hvor store ressurser man må sette av for selve videoarkivsystemet.
- Ventetid for brukere – hvor lenge må en bruker vente før vedkommende blir betjent?
- Systemgjennomstrømming – hvor mange brukere kan gå gjennom systemet (tjeneren) over tid?

I slike kømodeller ser man på et system fra et stort tidsperspektiv. Det ankommer forespørsler i en bestemt rate, forespørslene blir køet en stund før de kan betjenes og så forsvinner de fra systemet i det de er ferdig betjent, slik som skissert i figur 4.10. Her er det kun en oppgave som betjenes av gangen, dvs. vi har kun en båndstasjon i systemet, for enkelhets skyld, og vi opererer med verdiene A_{lib}, T_{lib}, etc.

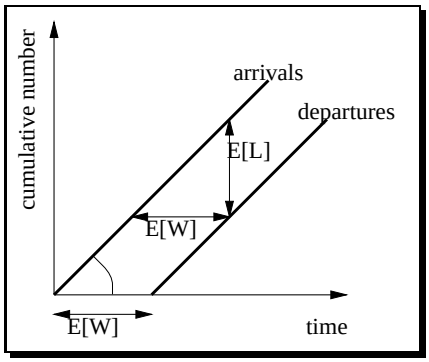


Figur 4.10: Skisse av et system med køing og en enkelt stasjon

Systemet er illustrert i figur 4.11, der man har to skråstilte parallelle linjer som viser akkumulert antall ankomster og avganger, og er vist i et stort tidsperspektiv slik at enkelthendelser ikke synes. Stigningstallet til linjene er lik ankomstraten (λ_{sys}). Den gjennomsnittlige ventetiden for betjening er $E[W_{sys}]$, og dette bestemmer avstanden mellom linjen for ankomster og den for avganger. Det

[1] Man kan evt. ha et eget nettverk mellom videopumper og båndarkiv.

gjennomsnittlige antallet forespørsler i systemet er $E[L_{sys}]$. Av figuren ser man at $E[L_{sys}] = \lambda E[W_{sys}]$. Denne formelen er kalt *Littles lov* (Dougherty 1990, Silberschatz og Galvin 1994). Vi har også at $E[L^q_{sys}] = \lambda E[W^q_{sys}]$, dvs. at lengden av selve køen er avhengig av ventetiden i køen og ankomstraten (Taylor og Karlin 1994).



Figur 4.11: Ankomster og avganger i et køsystem (Taylor og Karlin 1994)

Kømodeller beskrives ofte etter mønsteret $X/Y/z$, der X er distribusjonen av tiden mellom ankomster, Y er distribusjonen for ventetiden på forespørsler og z er antallet tjenere/prosessorer i systemet (Taylor og Karlin 1994), i vårt tilfelle vil dette bli antall båndstasjoner. X og Y kan blant annet anta følgende distribusjoner:

- M : hukommelsesløs (memoryless) – intervallet mellom hendelser er eksponensielt distribuert. Den kalles hukommelsesløs, fordi en ankomst ved tid t ikke påvirker sannsynligheten av fremtidige ankomster.
- G : generell distribusjon – intervallet mellom hendelser følger en vilkårlig statistisk distribusjon.
- D : deterministisk distribusjon – systemet «vet» hvor lang tid det vil være mellom hver hendelse.

Tiden mellom ankomster av forespørsler er bestemt av omgivelsene, men svært ofte følger den en eksponensiell distribusjon, det vil si at det totale antall ankomster over tid er poisson-fordelt, jf. diskusjonen om pålitelighet i avsnitt 4.1. Man kan derfor ofte anta at X er hukommelsesløs (M). Før vi gjør diskuterer behandlingstiden for en forespørsel (W_{sys}) antar vi følgende, for å gjøre diskusjonen enklere.

- Vi har en modell der vi kan beregne aksesstid mot en vilkårlig blokk, gitt at båndet er spolt helt tilbake. Eventuelt har vi målt alle slike aksesser og har dem lagret på harddisk. A_{lib} er dermed deterministisk, gitt at systemet vet hvilken video som blir forespurt.
- Vi har en god modell for veksletid i båndbiblioteket, eller vi har målinger for alle mulige vekslinger som er mulige å foreta i arkivet.
- Det ikke er særlig mange forespørsler mot samme bånd, slik at bånd må veksles hele tiden, slik at det som står i de to forrige antagelsene vil gjelde til enhver tid.
- Feil oppstår så sjelden at vi kan se bort fra effekten av dem. Overføringsraten for båndbiblioteket, T_{lib}, er det dermed deterministisk.
- Man har gjort diverse reservasjoner i systemet, slik som vist i beregningene for aksesstid og overføringsrate for systemet. Dermed blir $T_{sys} = T_{lib}$ og $A_{sys} = A_{lib}$.

Dette innebærer likevel *ikke* at tiden det tar å utføre en forespørsel (S_{sys}) er deteministisk, fordi:

- Systemet vet ikke *hvilket videomateriale* som skal overføres til harddisker før brukeren angir det.
- Systemet vet ikke *hvor mye* av videomaterialet som skal overføres før brukeren angir det i forespørselen.

Når systemet først har fått disse opplysningene, vil det være i stand til å finne gode estimater for betjeningstiden, S_{sys}. Dersom vi har en vilkårlig statistisk distribusjon, G, som passer brukernes forespørsler, får derfor man en såkalt $M/G/1$ *kømodell* for videoarkivtjeneren.

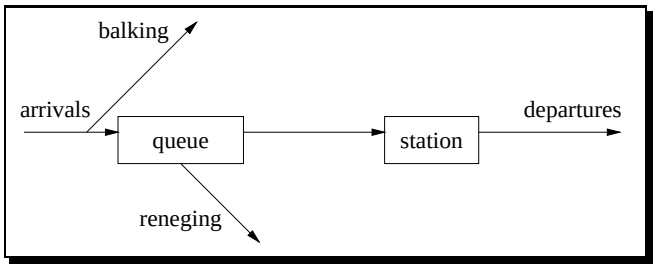
4.4.4 Utvidelser av kømodellen

Her ser vi overfladisk på noen aktuelle utvidelser av den kømodellen som ble presentert i forrige avsnitt. Vi ser på fire mulige utvidelser, som er aktuelle i en digital videoarkivtjener.

1. Ingen akseptering av nye forespørsler når køen for bruk av båndstasjonen er lang og/eller la brukerne kunne kansellere jobber som ligger i køen.
2. Behandle forespørsler forskjellig ut fra prioritet.
3. Overflyts-køing med to stasjoner. Man har to båndstasjoner, der den første tar i mot alle forespørslene den kan betjene, og sender resten til den andre, som tar imot forespørsler helt til den blir overbelastet. Når begge stasjonene er overbelastet, blir brukerne nektet adgang.
4. Flere båndstasjoner som kan behandle forespørsler i parallell.

Køing med nekting av forespørsler og kansellering

Dersom systemet nekter brukerne å legge inn forespørsler i køen fordi det er er overbelastet, kalles dette *balking* (Taylor og Karlin 1994), og er illustrert ved den oppadrettede skrå pilen i figur 4.12.

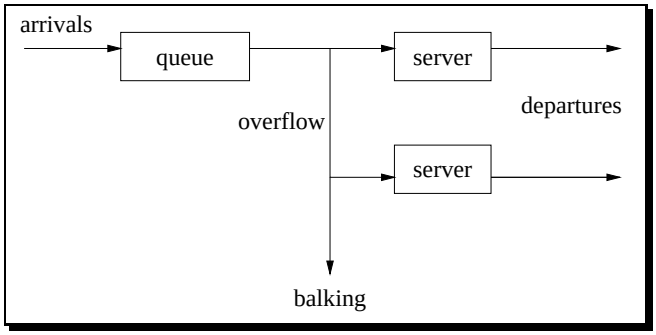


Figur 4.12: Skisse av et køsystem med nekting og kansellering

Dersom brukerne kan kansellere jobber som allerede ligger i køen, kalles dette *reneging* (Taylor og Karlin 1994). Dette er illustrert ved den nedadrettede pilen i figur 4.12. Sannsynligheten for at en bruker kansellerer en jobb i køen, kan også være avhengig av lengden på køen, men kan like gjerne skyldes hendelser i omgivelsene som systemet ikke har noen som helst kontroll over.

Overflytskøing

Dette er en enkel måte å organisere bruken av flere båndstasjoner på. Her ser vi på et eksempel med kun to båndstasjoner, der den ene primært utfører alle oppgaver som systemet får, og sender kun oppgaver videre til sekundær stasjon dersom den er opptatt. Dersom sekundær stasjon også er opptatt, *nekter* systemet å legge inn forespørselen i køen, akkurat som over.



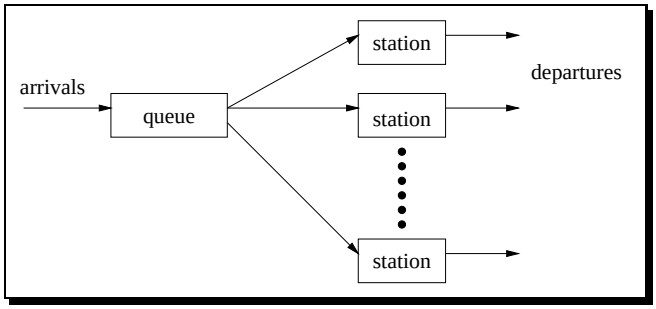
Figur 4.13: Skisse av et køsystem med overflyt og to stasjoner

Prioritetskøer

I prioritetskøer vil høyt prioriterte forespørsler få kortere ventetid enn de lavt prioriterte. Fordelen med en slik politikk er at forespørsler som er viktige kan gå forbi en lang kø av vanlige forespørsler, for å bli utført så snart som mulig. Dette kan være viktig i videoarkiver som skal betjene forskjellige grupper av brukere.

Køing og flere båndstasjoner

Kømodellen i avsnitt 4.4.3 gjaldt kun for en båndstasjon. Dersom vi har flere båndstasjoner, kan vi organisere køsystemet til å se ut som det i figur 4.14. Isteden for å ha en $M/G/1$ kømodell, har vi nå fått en såkalt $M/G/s$-modell, der s vil være antall stasjoner som kan benyttes i parallell.



Figur 4.14: Skisse av et køsystem med stasjoner i parallell

4.4.5 Oppsummering

En videoarkivtjener som lagrer data på digitale databånd er et system som kan bestå av en kontrollmodul, videopumper som skal overføre videostrømmer til brukerne via et høykapasitetsnettverk og en båndarkivmodul som tar seg av overføring av data mellom bånd og harddisk. Systemets oppførsel er avhengig av alle disse komponentene, selv om det i dette avsnittet er fokusert på båndarkivet. Det er skissert kømodeller som kan fungere som et grunnlag for statistisk analyse av et slikt system. Flere av parameterne i en slik modell er avhengige av omgivelsene.

4.5 Omgivelsene

Omgivelsene er viktige for en videoarkivtjener, fordi det er omgivelsene som legger premissene for om systemet er vellykket eller ei. Som nevnt flere steder i dette kapittelet er det også omgivelsene som bestemmer ankomstrate av forespørsler mot videotjeneren, hva slags video som skal hentes ut og hvor stor del av den brukeren er interessert i. Under er det beskrevet to mulige omgivelser for et system:

- Et museum ønsker å tilby direkte tilgang til en videotjener via en klientapplikasjon kjørende på en datamaskin i lokalet. Hvem som helst kan sette seg ned og hente ut det materialet vedkommende er interessert i. Her vil man antakeligvis få en eksponensiell fordeling mellom ankomster, slik som diskutert i forrige avsnitt.
- Hos en fjernsynsprodusent vil det være flere typer brukere av tjeneren, f.eks. arkivarer som skal registrere ytterligere informasjon om videomaterialet, nyhetsavdelinger som trenger bakgrunnsstoff, dokumentarteam som lager en reportasje, osv. I visse tilfeller vil man være i stand til å forutsi hva slags materiale man vil trenge en gitt dag, f.eks. materiale fra krigsutbruddet 9. april 1940, eller bilder i forbindelse med 17. mai. Slik informasjon kan brukes for å optimalisere systemutnyttelsen, ved at man passer på å utføre andre typer jobber så tidlig som mulig før hendelsen finner sted sted, slik at det er ledig kapasitet for de ekstra henvendelsene som man antar vil komme.

Det viktige her er at omgivelsene for en videoarkivtjener kan variere voldstomt fra installasjon til installasjon. I noen omgivelser kan man ha bruksmønstere som gjør at systemet bør optimaliseres, fordi brukernes ankomst ikke er tilfeldig.

4.6 Oppsummering

Overføringsrate og aksesstider vil være de viktigste karakteristikkene til en båndstasjon med ett bånd montert. Feilrater i forbindelse med lesefeil og stasjonsfeil kan påvirke disse karakteristikkene.

Et arkiv med flere bånd og en båndstasjon vil føre til at aksesstidene blir høyere. Gevinsten er (1) at kostnadseffektiviteten blir større, (2) at lesefeilraten blir mindre, fordi båndene er stort sett står lagret uten å bli utsatt for mekanisk slitasje i stasjonen og (3) at lagringskapasiteten blir større. Dessverre kan feilraten for selve arkivet bli høyere, fordi det må innføres mekanismer for transport av bånd mellom arkiv og stasjon, og disse kan feile.

Et båndarkiv med flere båndstasjoner fører til at båndbredden kan mot materialet som ligger lagret der økes. Hvor godt systemet takler feil på båndstasjoner, flyttemekansimer og lignende er svært avhengig av hvilken lagringsstruktur man har for dataene, dvs. om man benytter striping, redundans, osv.

Det går an å bruke statistiske modeller for å analysere hvordan en hel videoarkivtjener, med kontrollmoduler, videopumpe, båndarkiv og harddisker vil oppføre seg. Flere av parametrene som beskriver et slikt system er avhengige av omgivelsene for systemet.

Omgivelsene for en digitalt videoarkivtjener kan variere veldig fra installasjon til installasjon. I noen omgivelser kan det finnes brukermønstere som gjør det mulig å optimalisere tjeneren ytterligere.

Kapittel 5

Problemer og løsninger – alternativer og analyser

I dette kapittelet vil problemstillinger og mulige løsninger relatert til ESM bli diskutert og analysert. Det blir tatt utgangspunkt i fakta og analyse gitt i de foregående kapitler.

Diskusjonen er rettet mot en konstruksjon av ESM, men inneholder også alternative løsninger, som ikke vil behandles ytterligere i de senere kapitler. Følgende problemstillinger er behandlet i dette kapittelet:

5.1 Hvor skal ESM plasseres i Elvira-systemet?

5.2 Hvordan skal brukerne og administratorene komme i kontakt med ESM?

5.3 Hvordan og på hvilke måter skal ESM kommunisere med andre moduler i Elvira?

5.4 *Hvordan skal forespørsler om henting av videofilm håndteres?*

5.5 Hvordan finner ESM ut hvor lang tid forespørsler vil ta?

5.6 *Hvordan skal harddiskbufferet håndteres?*

5.7 *Hvordan skal data overføres?*

5.8 *Bør ESM ha databaser (på disk) med informasjon om innholdet i båndarkivet?*

5.9 I hvilken grad vil valg formater for lagring av mediadata påvirke ESM?

De fire fremhevede problemområdene er spesielt viktige i ESM, siden de vil utgjøre «kjernen» i systemet.

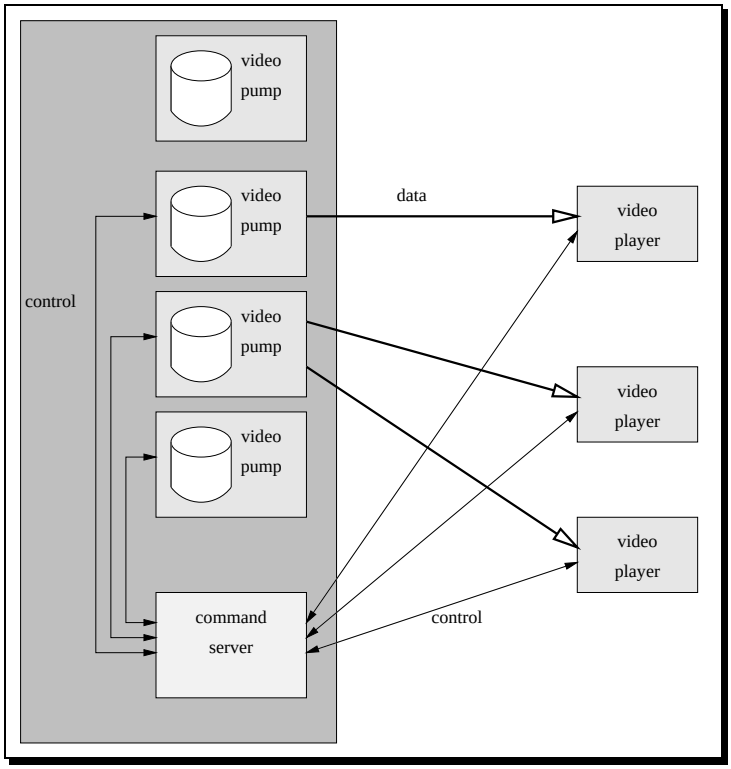
5.1 Beskrivelse av Elvira

Det er viktig å kjenne ESMs plassering i systemet, siden dette gir mye informasjon om hva slags problemer som må løses, særlig når det gjelder kommunikasjon og utveksling av data mellom moduler.

5.1.1 Kommandotjener

Kommandotjeneren, kalt Command Server[1] (CS), er hjernen i Elvira2. Overordnet struktur for Elvira2 er vist i figur 5.1, og der er CS tegnet inn. CS er en prosess som kjører på en av datamaskinene

[1] Denne prosessen ble i Elvira1 ble kalt Call Control.



Figur 5.1: Overordnet struktur for Elvira2 (Sandstå 1997)

som utgjør videotjeneren, og kontrollerer blant annet videopumper (VPer) på de andre maskinene. Det er også tenkt at CS skal styre ESM. Her følger en en beskrivelse av hovedoppgavene til CS (Sandstå 1997):

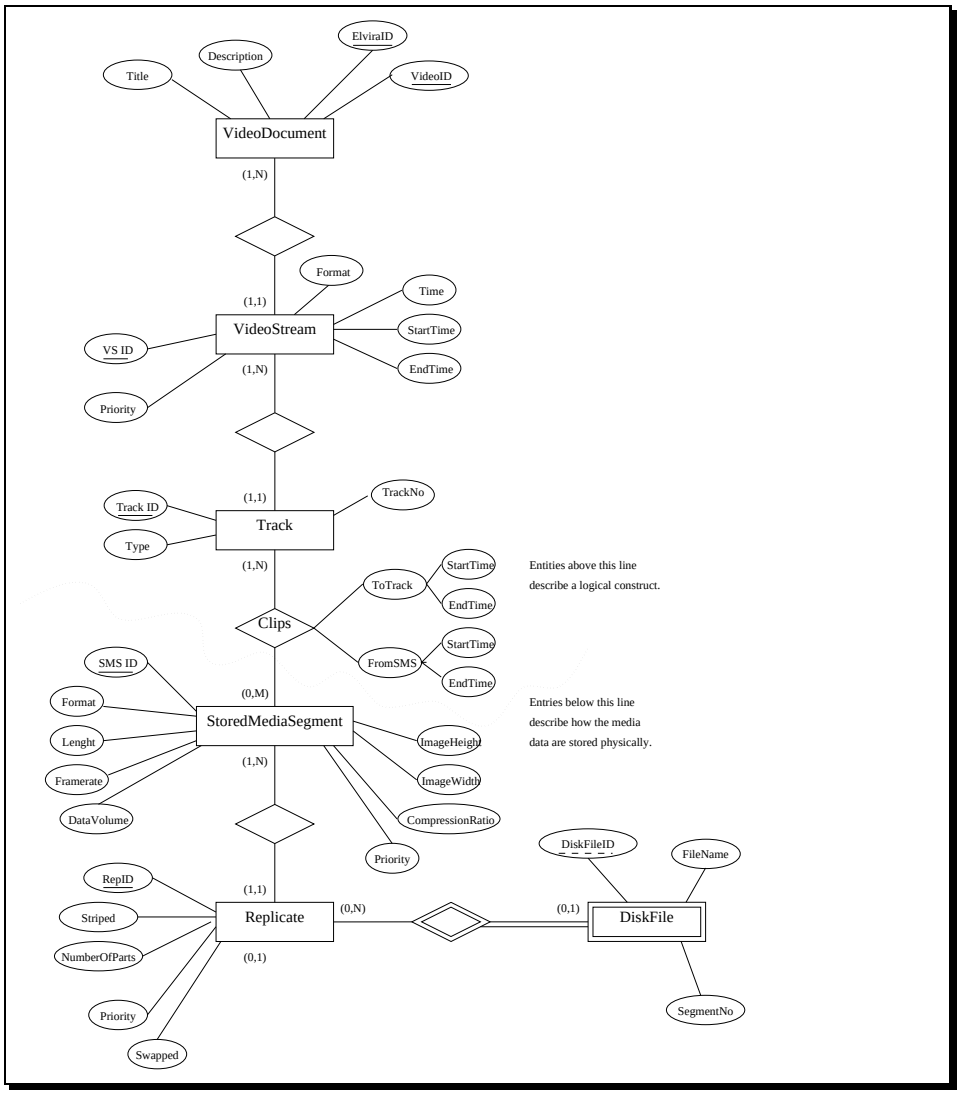
- Klienter henvender seg til CS når de vil bestille video fra Elvira.
- En henvendelse består av en fil-identifikator, navn på maskinen som klienten befinner seg på og to port-nummer Disse portene brukes for dataoverføring og kontroll.
- CS sjekker om Elvira har tilstrekkelig kapasitet til å utføre forespørselen, og om den har fått nok opplysninger til at dataene som skal overføres blir funnet. Dersom alt er i orden setter CS opp videopumper til å overføre data til klientens maskin.

5.1.2 Videopumper

Alle datamaskiner som skal delta i videoleveranser har en eller flere store harddisker med plass til mediadata. Disse maskinene har alltid en videopumpe-prosess (VP) kjørende. Videopumpenes oppgave er å overføre datastrømmer fra harddiskene til klientene som har forespurt video. I Elvira2 er pumpeprosessene flertrådet og fungerer slik:

- Det finnes en enkelt *kommando-tråd* som håndterer forespørsler fra CS. Her blir båndbredde allokert og frigjort, og strømmer startet og stoppet.
- Det kjører en *leveranse-tråd* for hver strøm som skal leveres til klienter. Disse benytter disk-trådene for aksess mot diskene og bufrer dataene for overføring via nettverket.
- Det er to *disk-tråder* for hver harddisk koplet til maskinen som videopumpen kjører på. Her blir de tilgjengelige ressursene fordelt og prioritering av diskaksesser foretatt.

5.1.3 Kataloghåndterer



Figur 5.2: Databasebeskrivelse for ECM

Katalogtjeneren, kalt Elvira Catalog Manager (ECM), er en enkel katalogtjener hvis oppgave er å holde oversikt over den fysiske lagringsstruktur for videomaterialet som befinner seg i Elvira. Database-skjemaet er vist i figur 5.2. Her følger en kort beskrivelse av tabellene i ECM (Sørensen og Sandstå 1996):

- Et videodokument er satt sammen av en eller flere ikke-overlappende videostrømmer. Et eksempel på et videodokument kan være en Dagsrevy-sending med flere innslag.
- En videostrøm består av en eller flere spor (tracks).
- Et spor (track) kan f.eks. være en MJPEG-video eller et lydspor med lyd samplet i CD-kvalitet.
- Spor og lagrede mediasegmenter (se neste punkt) er «limt sammen» av kutt-sekvenser (clips). Slike kuttsekvenser angir hvor et mediasegment av en viss lengde skal plasseres inn i et spor. Når mediasegmenter benyttes i flere spor, får vi såkalte «virtuelle» videodokumenter, som er avhengige av andre videodokumenters eksistens.

- Et lagret mediasegment (SMS) er en beskrivelse av en fysisk mediastrøm. Et slikt segment er beskrevet av rammerater, bildestørrelse, osv. Et mediasegment kan lagres ett eller flere steder, i en eller flere identiske replikater.
- Et replikat kan enten kun befinne seg på disk, på bånd eller begge steder.
- Replikater kan stripes ut over flere disker, og man må derfor dele det opp i diskfiler.

Følgende operasjoner kan gjøres på tabellene i katalogen:

- Sette inn poster.
- Slette poster.
- Endre poster.
- Liste ut alle poster i en tabell.
- Liste ut alle poster i en tabell som tilfredsstiller visse kriterier (søk).
- Rekursiv listing av all informasjon vedrørende en videostrøm.
- Rekursiv sletting av videodokumenter (ned til kutt-nivå)
- Rekursiv sletting av mediasegmenter (ned til diskfil-nivå)

5.1.4 Båndtjener

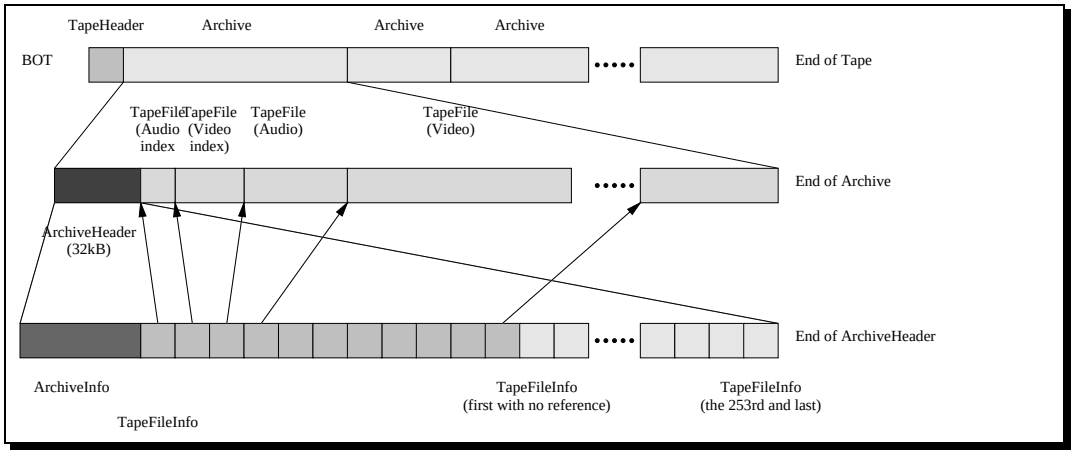
På IDI er det laget en laget en båndtjener (Sætre 1997), kalt Elvira Tape Server (ETS), som tilbyr bufret dataoverføring til og fra en båndstasjon. Grensesnittet mot brukeren er uavhengig av hva slags stasjon som blir benyttet, mens underliggende moduler er laget spesielt for Tandberg Datas TDC 4220 og MLR-1 båndstasjoner. Grensesnittet ble utviklet i samarbeid med veileder og undertegnede. ETS bruker følgende terminologi om det formatet som er valgt for lagring:

- Båndheader (tapeheader) er den første blokken på båndet. Her lagres informasjon som entydig identifiserer båndet, og gjør det mulig å sjekke at rett bånd er satt i stasjonen.
- Arkiv (archive) er en samling båndfiler som er relatert til hverandre, f.eks. lyd og bilde i en videostrøm.
- Båndfil (tapefile) er den minste enhet for lagring på bånd.
- Arkivheader er betegnelsen på de første blokkene i et arkiv. Her ligger det lagret informasjon om når arkivet ble laget, hvor stort det er og hvilke båndfiler som hører hjemme i arkivet (med pekere til dit filen begynner i arkivet).

En skisse av formatet for lagring på bånd er vist i figur 5.3. Her ser man hvordan et vanlig arkiv med 11 båndfiler er plassert på et bånd. Kort fortalt tilbyr ETS følgende brukergrensesnitt for operasjoner mot båndstasjonen:

- «Åpning» av arkiver, dvs. at båndstasjonen leter seg fram til et arkiv som er forespurt av klienten.
- Lesing av en bestemt fil (spesifisert ved filidentifikator).
- Lesing av et bestemt område på en fil (spesifisert ved startadresse og lengde).
- Skriving av et helt arkiv.
- Delt minnebuffer (av ringbuffertypen) for overføring av data. Dette innebærer at klienter av ETS må kjøre på samme maskin som denne. Låsefunksjoner er integrert i klient-grensesnittet.
- Angivelse identifikator for båndstasjon.

Lagring i båndbiblioteker vil kreve noen få utvidelser av dette grensesnittet i fremtiden.



Figur 5.3: Skisse av båndlagringsformat

5.1.5 Terminologi brukt i ETS og ECM

Det er på sin plass å avklare hvordan katalogstrukturen i ECM er relatert til lagringsformatet brukt i ETS. Begge steder har man beskrivelser av store datamengder (arkiver, videostrømmer) som består av en eller flere mindre deler (båndfiler, lagrede mediasegmenter). I konstruksjonen vil det derfor være aktuelt å bruke nettopp arkiver til å lagre videostrømmer, og båndfiler til å lagre mediasegmenter. Kort oppsummert:

- Et *lagret mediasegment* i ECM vil tilsvare en *båndfil* i ETS.
- En videostrøm i ECM vil tilsvare et *arkiv* i ETS.

5.2 Grensesnitt mot brukere

Grensesnittene mot brukerne er ESMs «ansikt mot verden». Det er imidlertid ikke sikkert at ESM bør tilby egne grensesnitt. Det kan være aktuelt å la all kommunikasjon med brukere gå gjennom en eller to prosesser som tar seg av dette for hele Elvira, f.eks. Command Server.

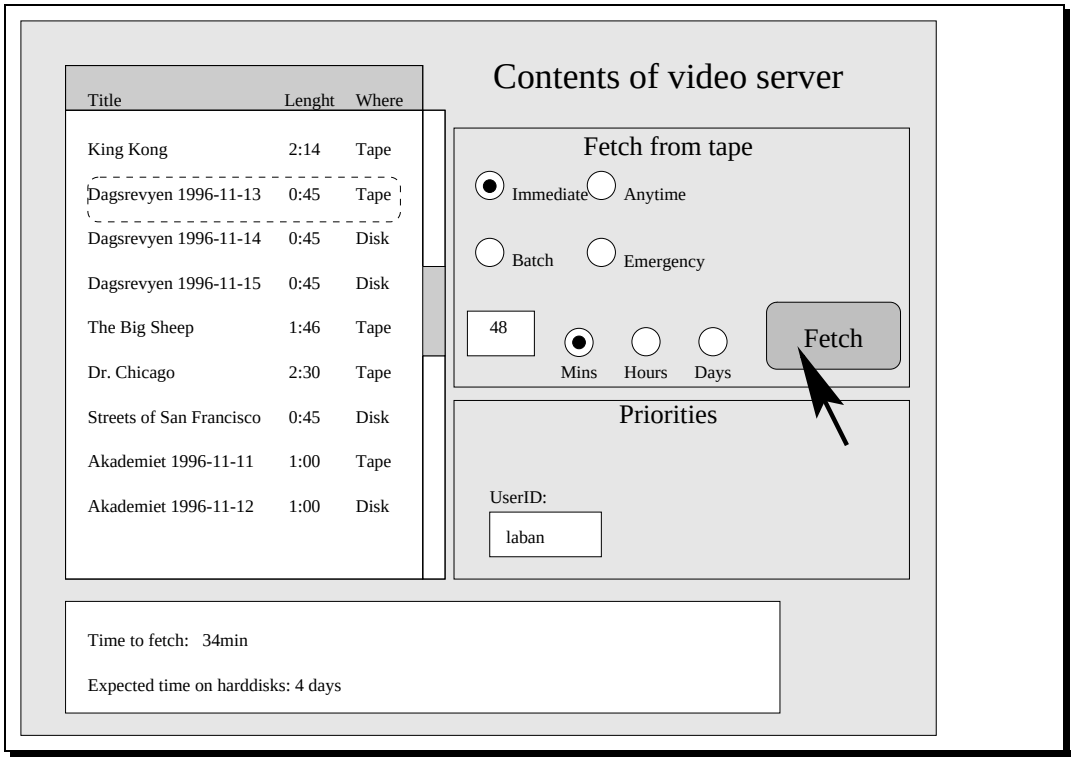
5.2.1 Grensesnitt mot sluttbrukere

Brukerne vil møte ESM enten direkte eller indirekte gjennom en eller flere brukergrensesnitt når de ønsker å få tak i videomateriale som kun befinner seg i båndarkivet.

Hvordan skal selve grensesnittet lages?

Hvordan selve grensesnittet mot brukeren lages, er ikke av så stor betydning for denne hovedoppgaven. Følgende alternativer er nok mest aktuelle:

- Vindusapplikasjon (for X) – kan implementeres i C++ ved hjelp av et grafikkbibliotek som f.eks. Qt (Troll Tech 1996) eller Motif.
- CGI-grensesnitt – kan implementeres ved hjelp av CGI++ i C++ (Sørensen 1996*a*). Krever endel arbeid med implementering.
- Java-applet – krever endel grensesnitt-kode slik at programmet som er skrevet i C++ kan snakke med java-grensesnittet.



Figur 5.4: Et mulig brukergrensesnitt mot ESM

De to siste alternativene er vel de som har mest appell blant dagens brukere. Som nevnt i avsnitt 3.2.3 har SGI benyttet seg av et www-grensesnitt med godt resultat. I figur 5.4 er et mulig grafisk brukergrensesnitt mot ESM skissert.

IDI har andre programsystemer for multimedia som etterhvert skal kunne benytte Elvira som tjener for videoleveranser. VideoSTAR (Hjelsvold 1995) og WebSTAR (Hetland 1996), som er en sterkt forenklet Java-versjon av VideoSTAR, er eksempler på dette. Det er derfor ikke sikkert at brukerne vil kommunisere direkte med Elvira/ESM i det hele tatt, men heller med en kontrollprosess i en av disse systemene.

Inngangsporter for brukere av Elvira

Hvilke inngangsporter skal Elvira ha for brukere? Skal hver modul ta seg av sin del av brukerkommunikasjonen, eller skal all kontakt med brukerne skje på en sentralisert måte? Det finnes altså to alternativer for ESM:

1. Distribuert: Direkte kommunikasjon med ESM.
2. Sentralisert: Indirekte kommunikasjon med ESM via CS og en sentral metainformasjonstjener.

Fordelen med sentraliserte løsningen er at antallet grensesnitt mot andre selvstendige Elviramoduler blir færre. Den sentrale metainformasjonstjeneren er introdusert fordi CS er en kontrollmodul som er optimalisert for å kun betjene videospillere. Denne skal jobbe raskt og uforstyrret og skal betjene så mange klienter som mulig. Dersom pågangen fra brukerne for å hente ut metainformasjon, all informasjon som ikke er mediadata, vil klientene som bruker videospillere bli skadelidende. Det vil derfor være svært aktuelt å ha en egen prosess, som gjerne kjører på en annen

vertsmaskin, som tar seg av alle forespørsler om metadata. Dette programmet vil blant annet bruke ECM og ESM for å svare på brukernes forespørsler om informasjon.

En annen fordel ved den sentrale løsningen er at all kommunikasjon med klienten enten går via Command Server eller den sentrale metadatabasetjeneren. Derfor blir blant mye enklere å kontrollere brukernes adgang til systemet.

Ulempen med en sentralisert løsninge at gjenfinning av feil kan bli vanskeligere under utvikling og drift, siden brukernes forespørsler bruker et ekstra «hopp» fra klient til ESM.

Hva skal brukeren spesifisere?

Når ESM skal hente data fra bånd, er det viktig at systemet ikke henter mer informasjon enn det brukeren har behov for, ellers vil mye tid være bortkastet. Man kan tenke seg følgende alternativer for en implementering av slik funksjonalitet, der valgmulighetene for brukerne øker gradvis:

1. Systemet leverer alle data som befinner seg i et arkiv (dvs. alle mediasegmenter knyttet til en videostrøm, for å bruke ECM-terminologi).
2. Brukeren angir hvilke typer og kvaliteter av materiale hun ønsker av videostrømmen som skal hentes ut, og ESM overfører kun de mediasegmenter som er i samsvar med det som er etterspurt.
3. Brukeren angir i tillegg hvilke intervaller av videostrømmen hun ønsker.

Alle alternativene har fordeler og ulemper. Dersom man for en videostrøm kun henter ut videodataene, som så lagres på harddisk, og man senere vil aksessere lyden (noe som ikke er helt usannsynlig), så slipper man å gjøre en ny forespørsel til ESM, fordi dataene allerede ligger i bufferet. På den annen side vil ESM spares for bortkastet overføring av data som aldri vil bli brukt ved å følge alternativ 2. Alternativ 3 vil føre til en ytterligere reduksjon i behov for båndbredde mot arkivet, siden mengden materiale som skal leses fra båndarkivet blir enda mindre. Ulempen er at ESM blir mer kompleks, fordi den må holde orden på hvilke videostrømmer som er hele, og hvilke som bare består av små biter når forespørsler ankommer systemet.

Brukernes påvirkning av ESM

Lagring på bånd må utvilsomt få visse konsekvenser for sluttbrukerne av systemet. Faktisk vil systemet sette visse krav til at brukerne oppfører seg rasjonelt, siden en bruker som bevisst går inn for å sabotere et system vil klare dette uten problemer. Det er imidlertid viktig at systemet gjør det så vanskelig for brukere å gjøre feil valg. Et godt grensesnitt som gir brukerne oversikt og mulighet for interaksjon vil være et bidrag til dette.

Tilbakemelding til brukerne

Det vil være fornuftig at ESM gir tilbakemelding til brukerne via brukergrensesnittene, slik at de kan reagere på den informasjonen de får. Et eksempel kan være en bruker som ber om å hente ut en film på et tidspunkt der systemet vil være hardt belastet. Det kan da være aktuelt for ESM å nekte å utføre denne forespørselen. Slik umiddelbar tilbakemelding er enkelt å legge inn i brukergrensesnittene diskutert tidligere.

Dersom ESM derimot ikke sitter i direkte kontakt med brukeren og det skjer en hendelse som vedkommende bør få vite, f.eks. at videodokumentet som brukeren forespurte nå er klart til å spille av, eller at en forespørsel har blitt kansellert på grunn av en nødsituasjon. Siden dette er hendelser som gjerne vil skje når brukeren ikke aktivt kommuniserer med Elvira/ESM, må andre former for kommunikasjon med brukeren tas i bruk. En enkel, og elegant løsning, ville være å lagre brukernes E-postadresser i en brukerdatabase. Brukeren vil så få en E-post som er generert av systemet, der

all nødvendig informasjon vedrørende avbrytelsen er gitt. En annen løsning, som kan brukes alene, eller sammen med E-post, er at brukerne har egne «postkasser» i Elvira, der de kan lese av status for egne forespørsler når de ønsker det.

5.2.2 Grensesnitt mot administrasjonsbrukere

Det kan tenkes flere forskjellige typer synlige grensesnitt som ESM vil ha mot administratorene.

- Kommandolinjebasert – svært enkelt å implementere. Kan bli omstendelig å bruke. Krever at administrator må logge seg inn på brukeren som tjeneren kjøres fra.
- X-baserte brukerapplikasjoner laget i C++ med f.eks. Motif eller Qt.
- Bruk av WWW og CGI-grensesnitt.
- Bruk av WWW og Java-applets.

Mye av diskusjonen i forrige avsnitt gjelder også for administrasjonsbrukere. Det er imidlertid slik at konklusjonene blir noe annerledes for administrasjonsbrukere. Årsakene er at administratorbrukere har:

- erfaring med ESM og har dermed bedre *oversikt* med hvordan de indre deler fungerer.
- har erfaring i bruk av UNIX og dermed ikke ingen problemer med å bruke simple brukergrensesnitt.[2]
- spesielle rettigheter i ESM som gjør at de kan «slå av systemet» med jevne mellomrom, slik at man kan foreta vedlikehold av Elvira, og man behøver derfor ikke å bekymre seg om å komme i konflikt med prosesser satt i gang av brukere.

Dette fører til at man stort sett kan nøye seg med å lage enkle kommandolinjebaserte grensesnitt mot ESM for administrasjonsbrukere. Ulempen er at administratoren må reservere tider der de får hele systemet for seg selv, noe som vil redusere effektiviteten til systemet.

5.2.3 Oppsummering og konklusjon

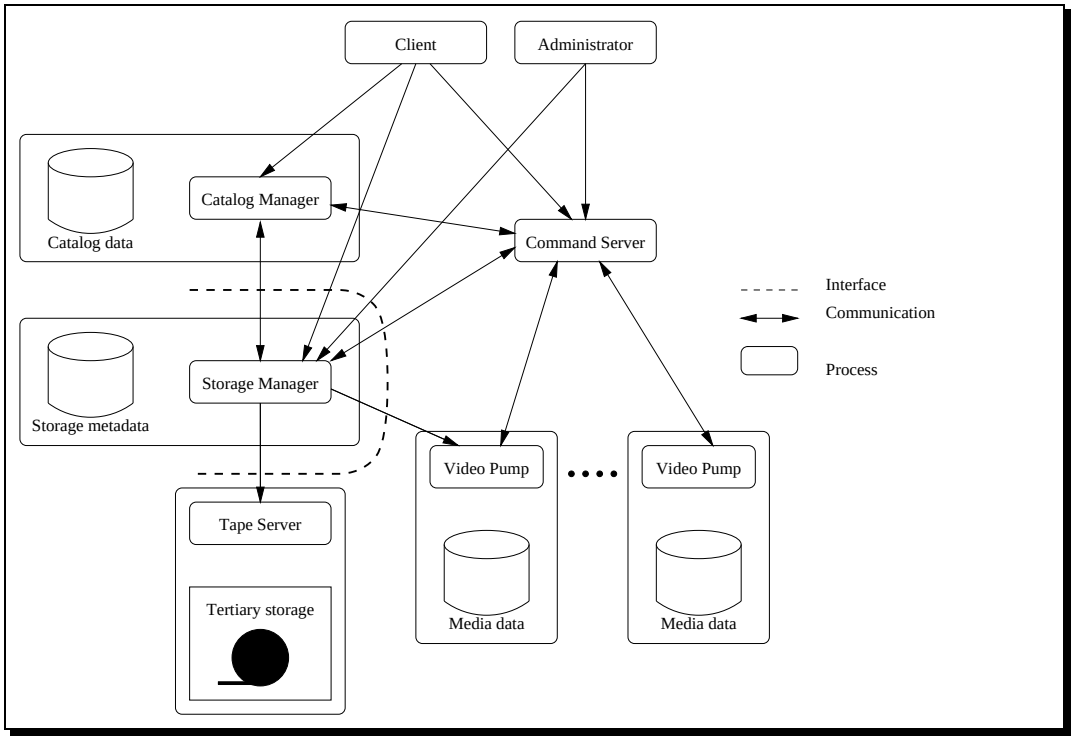
Grensesnitt mot sluttbrukerne bør gjøres informative og intuitive. Bruk av et WWW-grensesnitt med Java og/eller CGI er svært aktuelt. Administratorbrukere har helt andre behov, og kan nøye seg med (eller til og med foretrekke) enkle kommandolinjebaserte applikasjoner for vedlikehold.

5.3 Kommunikasjon

Som det går fram av avsnitt 5.1 så vil ESM være avhengig av andre moduler i Elvira, og vice versa. Dette innebærer at de må kommunisere med hverandre, slik som vist i figur 5.5.

Noen av modulene vil være sterkt knyttet til ESM og kjøre på samme vertsmaskin, slik som ESMs metadatabase og ETS. Andre moduler kan kjøre på andre maskiner knyttet til vertsmaskinen via et nettverk.

[2] faktisk er det nyttigere med kommandolinjebaserte programmer enn interaktive dersom de f.eks. skal startes med *cron(1)*, eller kjøres som batchjobber.



Figur 5.5: Mulige grensesnitt mellom ESM og andre moduler i Elvira2

5.3.1 Hvor mange kommunikasjonsgrensesnitt behøves?

I figur 5.5 ser man en skisse av en tenkt Elvira2 med en integrert ESM. Kommunikasjonen er vist med piler i retning fra klient til tjener. Som man kan se av figuren, *kan* være aktuelt for ESM å kommunisere med følgende moduler:

- Command Server (CS).
- Elvira Catalog Manager (ECM).
- Elviras Videopumper (VP).
- Diverse administrasjonsprogrammer, f.eks. program for innlegging av materiale i Elvira.
- Brukerapplikasjoner (se også neste avsnitt).
- Elvira Tape Server (ETS).

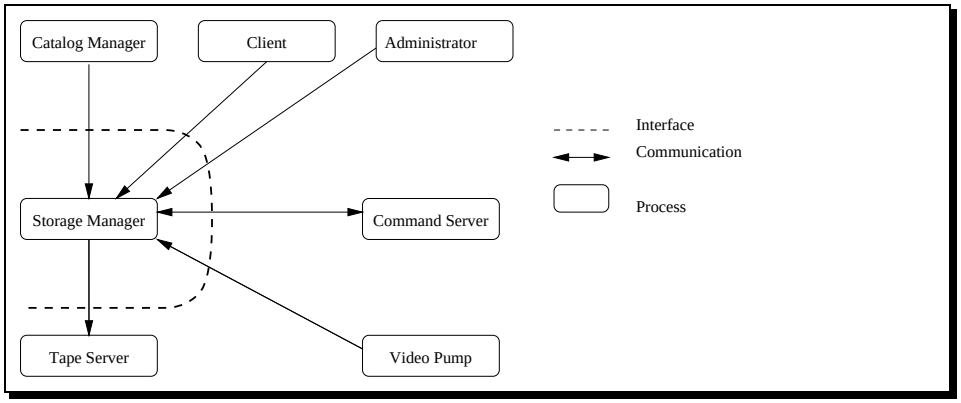
Dessverre innebærer så mange grensesnitt at det blir vanskelig å holde oversikt over hvem som egentlig bruker ESM, og dersom grensesnittene skulle konstrueres og implementeres, ville dette vært en stor jobb. Heldigvis går det an å begrense antallet grensesnitt ved å gjøre følgende antagelser:

- Sluttbrukere bør kun kommunisere med ESM via Command Server.
- ECM har intet behov for å blande seg bort i hva ESM driver med.
- Videopumpene trenger ikke å benytte seg av ESM i sitt arbeide.
- ETS trenger ikke å benytte seg av funksjoner i ESM.

Dette hjelper endel på situasjonen. De gjenstående grensesnittene er nærmeste umulig å eleminere. ESM må altså:

- tilby et klient-grensesnitt mot CS.
- benytte seg av klient-grensesnittet som CS tilbyr. Dette må gjøres når man f.eks. vil slette video som ligger i harddiskbufferet, eller gjøre klart for en overføring til videopumpene.
- benytte klient-grensesnittet som ECM tilbyr. Dette er beskrevet i dokumentasjonen av ECM (Sørensen og Sandstå 1996).
- benytte et databasegrensesnitt mot ESMs egne metadatabaser, slik som det som tilbys av f.eks. GDBM (Nelson og Gaumond 1995), eller Postgres (Fournier 1997).
- benytte et grensesnitt i videopumpene for å overføre data til dem når noe skal lagres i harddiskbufferet. Pumpene må da settes i «reversmodus». Dette grensesnittet blir altså kun brukt til ren dataoverføring. Kontrollen tar ESM og Command Server seg av.
- tilby et grensesnitt til administratorprogrammer, slik at man kan legge inn nytt materiale på bånd og hente ut informasjon og statistikk for ESM.
- benytte grensesnittet som ETS tilbyr (Sætre 1997).

ESM blir altså en *tjener* for CS, videopumper og administrator-applikasjoner, og en *klient* av ECM, CS og ETS, slik som vist i figur 5.6.



Figur 5.6: Grensesnitt mellom ESM og andre moduler i Elvira2

5.3.2 Kommunikasjon mellom prosesser

Under UNIX/SunOS5 eksisterer det flere flere metoder for å få prosesser til å kommunisere med hverandre. Denne delen er delt i to, fordi man vanligvis bruker forskjellige metoder for kommunikasjon mellom (1) prosesser som kjører på samme maskin og (2) prosesser som kjører på forskjellige maskiner.

Kommunikasjon på samme maskin

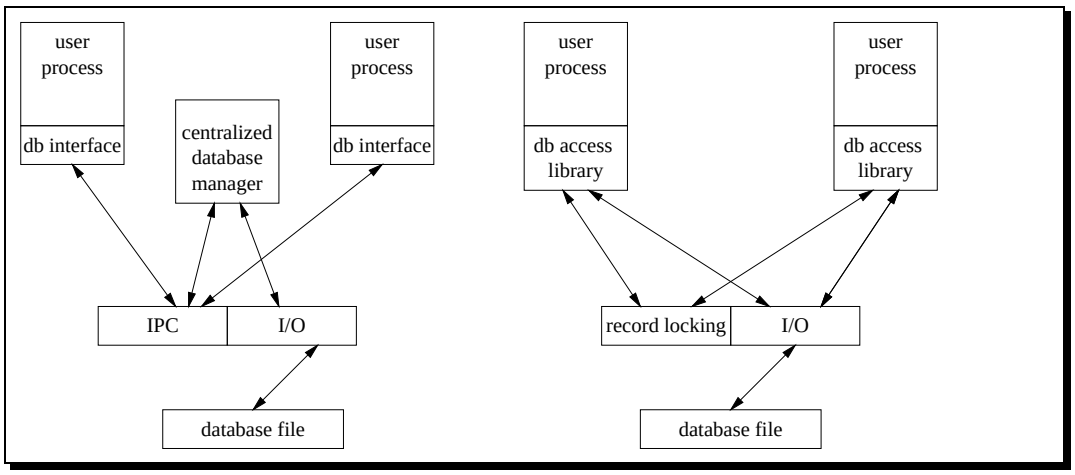
Når to prosesser kjører på samme maskin, kan man velge mellom flere mer eller mindre standardiserte kommunikasjonsmetoder (Stevens 1992, Robbins og Robbins 1996):

- Pipes, FIFOs (named pipes), stream pipes & named stream pipes – enkle former for køorientert overføring av data.
- Message queues, semaphores & shared memory – Ofte kalt System V IPC,[3] som er en mer avansert form for interprosesskommunikasjon, der to prosesser kan dele samme minneområde.

[3]IPC = interprocess communication.

Semaforer blir brukt for å låse deler av minnet ved lesing og skriving. Meldingskøer blir gjerne brukt for å sende kommandoer mellom prosessene.

Solaris støtter alle disse kommunikasjonsmetodene. I Elvira1 ble System V IPC brukt i stor grad, blant annet i klientenes videoavspillere og mellom pumpene og diskressursfordeleren. Et eksempel på bruk er gitt til venstre i figur 5.7, der vi ser to brukerprosesser som benytter seg av en databasetjener via IPC.



Figur 5.7: Prosesskommunikasjon via IPC og filsystem (Stevens 1992)

Prosesser på samme maskin kan imidlertid *også kommunisere via filsystemet*. Man må da bruke UNIX' støtte for fillåser (evt. lage sine egne rutiner for dette) for å sikre at data ikke blir ødelagt ved at flere skriver på en gang. En slik metode er vist på høyre side i figur 5.7, der man ser to brukerprosesser som begge er i stand til å jobbe mot databasefilen direkte.

På forskjellige maskiner

Når to prosesser kjører på to forskjellige maskiner har man følgende muligheter for kommunikasjon:

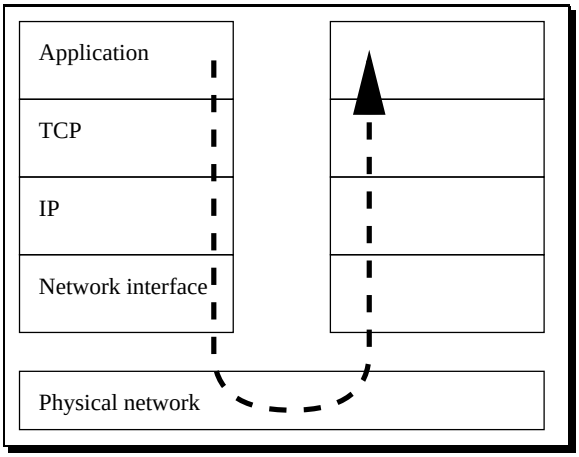
- Berkeley sockets
- System V TLI (transport layer interface)
- RPC (remote procedure call)
- CORBA (common object request broker architecture)

Sockets og TLI er enkle grensesnitt[4] mot internettprotokollens (IP) transportprotokoller (TCP og UDP). Sockets kan også benyttes mot andre nettverksprotokoller, som f.eks. Xerox NS (XNS). Berkeley sockets har vist seg å være den mest populære løsningen av de to (Stevens 1990). Sockets er benyttet i ECM (Sørensen og Sandstå 1996) og i Elvira (Langørgen 1994, Sandstå et al. 1996).

RPC er en metode som Sun utviklet for å lage funksjonelt orienterte distribuerte systemer. RPC fungerer ved at man lager en spesifikasjonsfil der man angir funksjonene som skal kunne kalles fra andre maskiner. Deretter brukes f.eks. UNIX-programmet *rpcgen* (Robbins og Robbins 1996) for å generere den koden som er nødvendig for at klient og tjener skal kunne kommunisere via nettverket. RPC er blant annet brukt for å implementere Sun NFS (network file system).

[4]Enkle i den forstand at de ikke tilbyr avanserte abstraksjoner for kommunikasjon. Det er nok av ting som kan gå galt med sockets og TLI også.

CORBA er en avansert metode for å lage objektorienterte distribuerte programsystemer, dvs. lage programmer der objektorienterte systemer kan kalle metoder i objekter som kjører på andre maskiner (Appelbaum et al. 1997). På samme måte som for RPC må man spesifisere de distribuerte objektene i et eget spesifikasjonsspråk (IDL) som det så blir generert kildekode fra. Det kreves mye arbeid å sette seg inn i CORBA, samtidig som behovet for kommunikasjon i ESM er av en forholdsvis enkel karakter. Det vil være «overkill» å benytte CORBA i ESM. En annen ulempe med CORBA er at den ikke følger med operativsystemer og at det ikke finnes gratis implementasjoner. Dette innebærer ekstrakostnader.



Figur 5.8: Dataenes vandringsvei fra sender- til mottaker-applikasjon

Prosesser som kjører på samme maskin kan selvsagt også benytte seg av de metodene som er vist her. Det er imidlertid to grunner til å la være å gjøre dette. For det første er det umulig å bruke delt minne, slik at man må sette av ekstra plass til bufring av data. For det andre, så er slik kommunikasjon tregere enn IPC spesialisert for kommunikasjon på samme maskin, siden nettverksoverføringer krever at dataene må gjennom flere lag med forskjellig koding før de blir overført, og så må de pakkes opp igjen på mottakersiden, som vist for TCP/IP i figur 5.8.

5.3.3 Protokoller

Dersom man ikke velger å bruke kommunikasjonsmetoder som har innebygde protokoller, må man selv sørge for slike. Det finnes minst to mulige måter å spesifisere en protokoll på, nemlig:

- å bruke Abstract Syntax Notation One (ASN.1) i spesifikasjon og bruke et program som oversetter spesifikasjonen til rutiner for protokollkoding (Stallings 1994) .
- å bruke en egen beskrivelse og egne rutiner for protollkoding.

ASN.1 er forholdsvis stort og komplisert å sette seg inn i, men har den fordelen at man kan bruke et verktøy kalt Snacc (Sample og Joop 1995) for å generere kildekode i C++ for å kode/dekode meldinger. En annen fordel er at kodingen er minimal, dvs. at man bruker et minimalt antall bits for å identifisere de forskjellige dataelementene man sender via protokollen.

Egne beskrivelser bør gjøres i på en semiformell måte, f.eks. ved bruk av BNF-lignende syntaks (Sethi 1989). På samme måte som i Snacc (Sample og Joop 1995) bruker en trelignende datastruktur der man putter inn informasjonen som skal overføres. Deretter kaller man en metode som går gjennom denne strukturen og genererer en strøm av tegn (enten binær eller tekstlig). Denne strømmen sendes over nettet og mottakeren kaller en metode for å bygge opp treet fra datastrømmen. Mottakeren går så gjennom treet og handler ut fra det som ligger lagret der.

Figur 5.9: Enkel forespørselskø

Undertegnede har prøvd begge metoder (Sørensen 1996*b*, Sørensen og Sandstå 1996) og har kommet fram til at når behovene for kommunikasjon er forholdsvis enkle og kravet til hastighet ikke er svært strengt, er det like greit å bruke egenlagde protokoller.

5.3.4 Oppsummering

ESMs plassering i Elvira kan kort oppsummeres slik:

- ESM blir en tjener for CS, videopumper og administrator-applikasjoner.
- ESM blir en klient av ECM, CS og ETS.

Kommunikasjon mellom prosesser vil gjøres med System V IPC når prosessene kjører på samme vertsmaskin og med sockets mot TCP/IP når de befinner seg på forskjellige maskiner. Enkle egenlagede protokoller vil bli benyttet.

5.4 Ressursfordeling

I et arkiv som har en stor mengde multimediadata lagret på bånd og et rimelig antall brukere, vil det være uunngåelig med konflikter om ressurser oppstår, i dette tilfellet om tilgang til materiale i båndarkivet.

Den enkleste form for fordeling er at brukerne får beskjed når båndarkivet er ledig. Brukerne sender så en forespørsel til systemet om å hente en spesifikk film. Dersom flere brukere sender en forespørsel på samme tid (innen et bestemt tidsintervall), foretar systemet et valg basert på prioriteten av forespørslene. Denne løsningen er enkel, men ikke særlig god. Den krever at brukerne hele tiden følger med på hva som skjer med båndarkivet og dette er uholdbart i de fleste tilfeller.

Ved å introdusere en ressursfordeler vil brukerne slippe å følge med på status for båndarkivet. Systemet må tilby mulighet for å legge inn forespørsler i en kø. Når systemet er klart til å utføre en forespørsel, blir en av forespørslene tatt ut av køen. Systemet har en ressurfordeler som ordner forespørslene i køen slik at båndarkivet blir best mulig utnyttet. Brukerne angir f.eks. når de ønsker at filmen skal være i harddiskbufferet. Jo mer informasjon systemet får av brukerne, jo mer nøyaktig kan ressursfordelingen foretas. Som nevnt i avsnitt 5.2 er det viktig at brukerne opptrer rasjonelt, dvs. at de angir de verdier som er passende for behovet de har.

I dette avsnittet presenteres to alternative måter å fordele ressursene i systemet på. Det første alternativet baserer seg på én kø, der alle forespørsler blir lagt inn. Det andre alternativet gir mulighet for flere køer, der hver kø blir behandlet individuelt av systemet. I begge modeller antar man at båndarkivet kun har én båndstasjon, for enkelhets skyld. Behov for modifikasjoner på grunn av (1) bruk av flere båndstasjoner i et arkiv, (2) mulighet for kansellering og (3) avvisning av forespørsler er presentert til slutt.

5.4.1 Enkel forespørselskø

Ressursfordeleren må ha visse data inn fra brukerne for at den skal kunne fatte fornuftige avgjørelser. Det er en grense for hvor mye informasjon en bruker er villig til å gi for hver gang hun ønsker

å hente en film fra tertiærlager, derfor bør systemet prøve å få tak i mest mulig informasjon uten å være avhengig av brukeren, se avsnitt 5.5. Her antas det at det er kun parameterne som skiller forespørsler fra hverandre, ellers er de like.

Parametre

Når brukeren bestiller en film, så vil hun angi visse verdier for en del attributter som (forhåpentligvis) samsvarer med de reelle behov hun har. Hver forespørsel vil få en *profitt-verdi* som er avhengig av parametrene som beskriver en forespørsel:

- En *film-identifikasjon*.
- *Tidsfrist* for når filmen bør være inne i systemet.
- *Toleranse* for overskridelser av tidsfrist.
- Det ønskede *tidsintervall* som man ønsker å se av filmen.
- Hvilken *kvalitet* filmen skal ha.
- En *bruker-identifikasjon*.

Disse profittverdiene brukes så i ressursfordelingen for å veie forespørsler opp mot hverandre (Brassard og Bratley 1987). Dersom arkivet kan utføre forespørselen, later man som om det *tjener* denne verdien. Optimaliseringen går derfor ut på å få mest mulig profitt ut av et arkiv over et gitt tidsintervall. Ut i fra disse parameterne og informasjonen som er hentet fra metadatabasene (se avsnitt 5.5) bestemmes de variable som vil brukes av algoritmen når den skal bestemme ressursfordelingen:

- i – Forespørselsnummer.
- d_i – Tidsfrist.
- t_i – Hvor lang tid operasjonen vil ta.
- p_i – Profittverdi som tildeles båndarkivet dersom forespørsel i kan utføres.
- D_i – Mengde data som skal overføres.

Tidsfristen blir satt til det brukeren spesifiserer, verken mer eller mindre. Toleransen for forsinkelse vil kun bli tatt i bruk dersom det er nødvendig. Hvor lang tid selve overføringen vil vare (t_i), er avhengig av hvor mye materiale som skal overføres og hva slags egenskaper båndarkivet har (se kapittel 4, side 29 og avsnitt 5.5).

Å gi en profittverdi til en forespørsel er egentlig en svært vanskelig oppgave, fordi man må ta hensyn til politikken i den organisasjonen som videotjeneren skal betjene. Det enkleste er å sette opp en rekke parameterne som systemet bør hensyn til, men la administratoren bestemme hvordan de forskjellige parameterne skal vektes. Følgende faktorer *kan* påvirke en profittverdi i ESM:

- Brukerens prioritet.
- Hvor mye brukeren er villig til å betale[5] for tjenesten.
- Hvor lang tid på forhånd brukeren bestiller filmen. Det skal lønne seg å være tidlig ute. (Dette er en parameter som typisk vil utelates i visse organisasjoner.)
- Om noen andre har bestilt filmen så vil dette telle (svært) positivt. Dette er nærmest et unntakstilfelle som må utredes nærmere senere.
- Kvalitet koster. Det skal lønne seg å bestille lavkvalitets video/audio.
- Kvantitet koster. Det skal lønne seg å bestille korte tidsintervaller.

[5] dette kan f.eks. skje i en bedrift som benytter internfakturering

Verdien p_i må normaliseres mot tiden for å få «profitt per tidsenhet» ($P_i = \frac{p_i}{t_i}$), slik at algoritmen velger ut de forespørslene som gir båndstasjonen optimal profitt per tid. Det vil si at de to siste punktene egentlig er tatt med i beregningene allerede.

Allokering av forespørsler i kø for én båndstasjon

For det første bør algoritmen føre til at båndstasjonen utnyttes maksimalt. En båndstasjon har forholdsvis høy enhetskostnad og utnyttelsesgraden vil derfor være en viktig faktor ved kostnadsberegninger. For det andre bør algoritmen være i stand til å kunne gi umiddelbare svar tilbake til brukeren om systemet har kapasitet til å håndtere denne brukerens forespørsel.

Det finnes flere algoritmeeksempler i litteraturen på jobbfordeling for en enkelt «prosessor» (i vårt tilfelle båndstasjonen i arkivet) der hver oppgave tar én tidsenhet (Brassard og Bratley 1987, Cormen et al. 1990, Horowitz og Sahni 1978). En slik algoritme må utvides til å gjelde flere båndstasjoner og oppgaver som har forskjellig utføringstid. Dette gjøres svært enkelt ved å se på profitt per tidsenhet, dvs. $P_i = p_i/t_i$. Da vil man sikres en optimal utnyttelse av arkivet også når utføringstidene er variable. I tillegg må det legges inn en ekstra sjekk på at en oppgave som skal inkluderes ikke overlapper andre oppgaver.

5.4.2 Flere forespørselskøer

Brukere av et stort system har vanligvis varierende behov for tjenester som systemet tilbyr. Ved å introdusere forskjellige typer forespørsler i systemet, kan brukerne velge hvordan systemet skal behandle forespørselen ut fra typen tjeneste som ønskes.

Forespørselstyper og køer

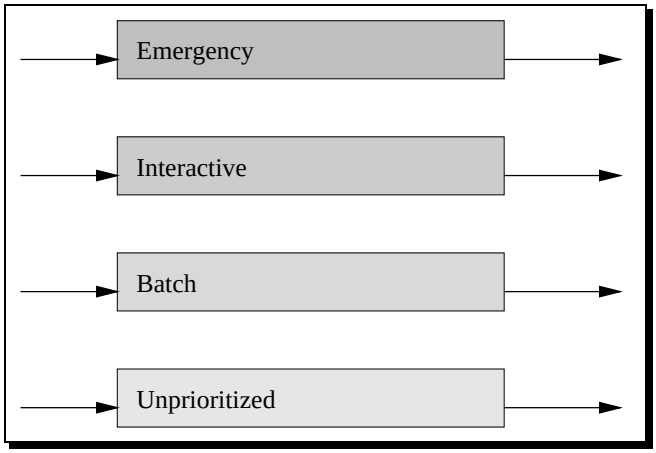
For ESM kan det være naturlig å dele opp forespørsler i kategoriene vist i figur 5.10. Et lignende system er foreslått for CPU-fordeling i boken *Operating System Concepts* (Silberschatz og Galvin 1994). Hver type forespørsel har regler for hvordan de skal (1) relatere seg til forespørsler av andre typer og (2) hvordan den interne fordelingen av forespørsler av samme type skal foregå. Den siste typen fordeling er ikke annet enn det som ble diskutert i foregående avsnitt. En vanlig bruker kan f.eks. få muligheten til å velge mellom følgende fem typer forespørsler:

- E-jobb – Nødoppgave (emergency). Bare svært prioriterte brukere er autorisert til å legge inn slike oppgaver i ESM.
- B-jobb – Garantert henting innen et visst tidspunkt (batch). Kun for prioriterte brukere.
- I-jobb – Interaktiv henting av data fra bånd. Kun for prioriterte brukere.
- U-jobb – Uprioritert innhenting. Kan brukes av alle.
- A-jobb – Reservert tid for administrasjon av ESM. Kun for administratorbrukere.

Tidsintervaller

Det virker fornuftig å dele opp tidsaksen i intervaller, for å ha muligheten til å fordele ressurser rettferdig mellom de forskjellige forespørselstypene. Når man vet hvor lang tid det vil ta å utføre forespørslene, kan man holde regnskap over hvor stor prosentandel av en spesiell type forespørsel som legger beslag på tidsintervallet. Ved å sette grenser for hvor mange prosent en type forespørsler kan utgjøre av et tidsintervall, har vi skapt en mekanisme for fordeling av jobber.

Hvilke typer jobber bør det legges inn slike begrensninger for? Som det går fram over, er det bare A og B-jobber vil bli utsatt for slike grenser. E-jobber må av naturlige grunner få lov til kunne fylle



Figur 5.10: Flernivå forespørselskø

100% av et tidsintervall. I og U-jobber vil kun fordeles i det tidsintervallet som systemet befinner seg i, og vil kun begrenses av hvor mange forespørsler av de andre typene som allerede ligger inne i kjøreplanen. Lengden på tidsintervallene bør:

- være så lange at en hvilken som helst innhenting kan foregå innenfor intervallets grenser. Så lenge alle forespørsler kan utføres innen et tidsintervall, kan man abstrahere bort ressursallokeringen innenfor dette intervallet.
- være så korte at B-jobber ikke kan dominere bruken av båndstasjonen over lengre tid.

Dersom man tenker seg en intervall-lengde på 10 minutter, vil man helt klart få et problem dersom man ønsker seg en film som f.eks. tar 300MB, og overføringsraten er 1.5 MB/s. Det vil da ta 20 minutter å hente filmen fra båndet, og man må «dele opp» jobben over flere tidsintervall for å hente inn filmen.

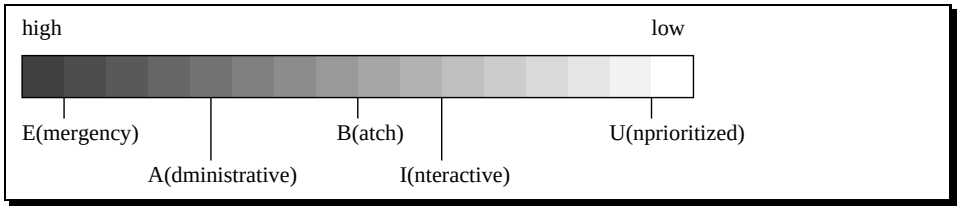
I det andre ytterpunktet kan man tenke seg et tidsintervall på 24 timer. Fordelen er at alle filmer av realistisk størrelse kan hentes ut i løpet av intervallet. Ulempen er at dersom man bruker en «ivrig» metode for å allokere B-jobber i et tidsintervall, så risikerer man at båndstasjonen kan være opptatt med å betjene slike jobber i flere timer før andre slipper til.

Behandling av køer

Når man har flere køer, må man ha en politikk for å velge mellom jobber fra de forskjellige køene. Et eksempel på en slik politikk kan kort oppsummeres slik[6]:

- E-jobber legges alltid inn, og kaster ut alle andre jobber som kolliderer med tidsintervallet E-jobben krever. Ingen andre jobber kan skyve ut E-jobber av kjøreplanen.
- B-jobber kan registreres framover i tid (innen rimelige grenser). Systemet aksepterer nye B-jobber i et bestemt tidsintervall så lenge ikke den samlede tiden av B-jobber overstiger en spesifisert prosentandel av tilgjengelig tid i intervallet. Systemet aksepterer heller ikke nye B-jobber i et nåtidig tidsintervall.
- I-jobber kan bare legges inn i nåtidig tidsintervall.
- U-jobber blir køet (FCFS) og prosessert når systemet har ledig kapasitet.

[6] og en utførlig beskrivelse finnes i avsnitt 6.2.1



Figur 5.11: Grad av «kostnad» på forskjellige jobbtyper

I tillegg kan det være aktuelt å ha mulighet for å legge inn administrasjons-jobber, der normale operasjoner mot båndarkivet vil stoppe opp, slik at nye filmer kan bli lagt inn. Det er imidlertid ikke alltid nødvendig med noe slikt. I et stort system vil man ha en såpass stor innstrømming av nytt materiale at det er aktuelt å reservere en eller flere båndstasjoner i arkivet for lagring av materiale. I et eksperimentelt system har man såpass lite (og kontrollert) trafikk at administrator kan legge inn filmer som E-jobber isteden (altså bryte all vanlig trafikk). Tidsintervallene kan være av tre forskjellige typer:

- Fremtidig – et tidsintervall der laveste grense har en verdi høyere enn nåværende systemtid. Her kan bare A, B og E-jobber allokeres.
- Nåtidig – et tidsintervall som inneholder nåværende systemtid. Her kan bare E, I og U-jobber allokeres.
- Historisk – et tidsintervall der øvre grense er lavere enn nåværende systemtid. Selvsagt kan ingen jobber allokeres her, men disse intervallene kan arkiveres for senere analyse av system-oppførsel.

I et system som fakturerer brukerne, kan det f.eks. være fornuftig at man betaler ekstra for forespørsler som (1) er spesielt ressurskrevende, eller (2) må behandles i løpet av kort tid. En annen mulighet er at kun prioriterte brukere har adgang til å legge inn I- og B-jobber. En foreslått gradering av prioriteter mot kostnad er gjort i figur 5.11.

Køing innen tidsintervall

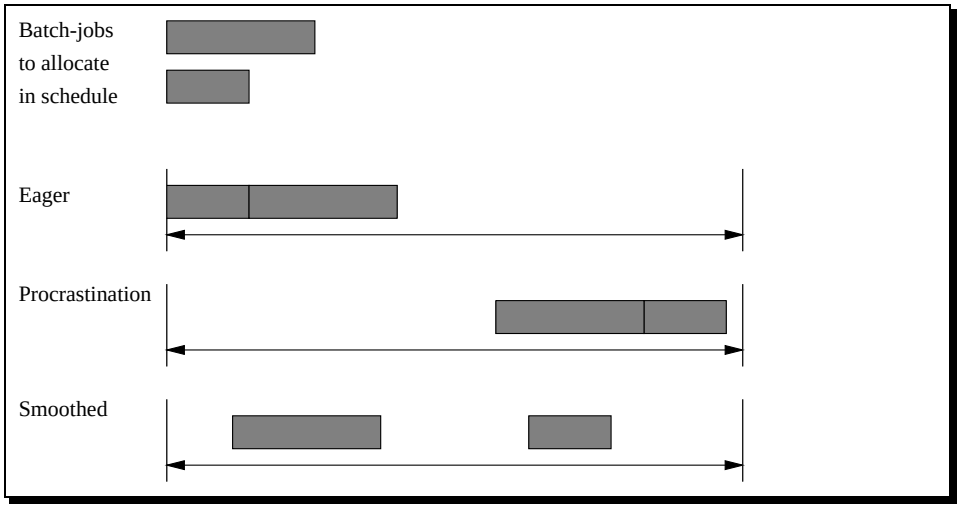
Ved å dele opp kjøreplanen i tidsintervallet, har man kun løst problemet med hvordan man prosentvis skal begrense antallet B-jobber i systemet over en viss tid. Hvordan man skal legge jobbene etter hverandre i et tidsintervall er fortsatt uløst.

Kjøreplanen for et tidsintervall behøver ikke å behandles før systemet begynner å arbeide med det (dvs. at tidsintervallet går fra fremtids- til nåtids-status).

Når systemet går fra et tidsintervall og over i et nytt, vil det måtte bestemme seg for når B-jobbene skal kjøres.[7] Så lenge alle B-jobbene blir utført før intervallets utløp, står ressursfordeleren fritt når det gjelder å sette dem opp på kjøreplanen. Man kan tenke seg følgende scenarier, som også er illustrert i figur 5.12:

- B-jobber utføres så raskt som mulig, og så slipper I og deretter U-jobbene til. Fordel: kontinuerlig ledig plass etter B-jobbene. Ulempe: systemet vil være svært opptatt i begynnelsen av hver periode, så lenge det jevnlig kommer inn B-jobber i ESM.
- B-jobber utsettes så lenge det (1) er tid igjen til å utføre dem og (2) systemet er opptatt med I og U-jobber. Fordel: fleksibilitet. Ulempe: systemet kan bli svært opptatt mot slutten av intervallet, dersom det har vært stor pågang av I og U-jobber tidligere.

[7] A og E-jobber er allerede spikret, og det er ikke rom for å endre tiden disse skal kjøres på.



Figur 5.12: Ressursplanlegging innen tidsintervall

- B-jobber fordeles utover perioden, slik at lasten fordeler seg rimelig jevnt utover tidsintervallet. Fordel: ingen perioder hvor systemet kun jobber med B-jobber. Ulempe: oppstykking av ledige perioder i små intervaller, som kan føre til at I og U-jobber av en viss lengde rett og slett ikke får plass.

5.4.3 Køing på system med flere stasjoner

De to alternative metodene for ressursfordeling nevnt hittil har kun vært for et system med en enkelt båndstasjon. Når man introduserer muligheten for flere båndstasjoner, slik vist som i avsnitt 4.3 på side 35, vil det bli vanskeligere å oppnå optimalisert bruk av ressursene. En naiv metode vil være å la en båndstasjon være primærprosessor. Når denne ikke kan ta imot en jobb, fordi den ikke passer inn i kjøreplanen, sendes jobben videre til neste båndstasjon.

Ved striping på bånd, der alle båndstasjoner deltar, vil problemet reduseres til køing på en stasjon, siden alle stasjonene vil være opptatt med den samme jobben, dog i kortere tid.

5.4.4 Kansellering

En bruker må også være i stand til å kansellere forespørsler, slik at systemet ikke blir belastet med å utføre henting av data som aldri vil bli brukt.[8]

Det kan også skje at ressursfordeleren må kansellere jobber som allerede er akseptert. Dette kan blant annet skje dersom det ankommer en nød-forespørsel i systemet beskrevet i avsnitt 5.4.2, og det ligger jobber i systemet som må skyves tilstede. I slike tilfeller bør de berørte brukerne få beskjed via en eller annen asynkron meldingsmekanisme, som f.eks. E-post.

5.4.5 Avvisning

Systemet vil måtte avvise jobber som ikke passer inn i kjøreplanene.[9] Det er imidlertid flere alternative måter å gi brukeren beskjed om dette på:

- Systemet sier nei, og brukeren må lage en ny forespørsel helt fra begynnelsen av.

[8] I kø-teorien kalles dette *reneging* (Taylor og Karlin 1994).
[9] Dette kalles *balking* i kø-teori (Taylor og Karlin 1994).

- Brukeren får tilbake informasjon om når en forespørsel av den typen hun bad om tidligst kan utføres av ESM.

Det er rimelig åpenbart at det siste alternativet kan føre til begrenset antall forespørsler mot ESM i travle perioder, siden brukerne nå vet hvor de bør legge inn en jobb.

5.4.6 Optimalisering av forespørsler mot samme bånd

Det som imidlertid kan gi en viss effektivisering, er om ressursfordeleren legger jobber som aksesserer samme bånd etterhverandre, slik at man unngår unødvendige skifter av båndkassetter. Dette er en forholdsvis enkel test som kan gjøres når ressursfordeleren mottar nye jobber. Det å fremskynde jobber slik at de kan samkjøres med andre jobber mot samme bånd har imidlertid den ulempen at de kan føre til økt belastning på harddiskbufferet.

Når det gjelder effektive algoritmer for fordeling av jobber på bånd, så har Silberschatz og Hillyer (1996*b*) testet ut flere algoritmer for aksesser mot DLT-bånd, som har omtrent de samme egenskaper som QIC når det gjelder søking. De avanserte algoritmene er mest effektive når man har et stort antall forholdsvis små filer på et bånd og så kan få flere samtidige forespørsler om forskjellige filer. I et videoarkiv vil det være forholdsvis få og store filer på et bånd. Det er derfor begrenset hvor mye en effektiv lokaliserings-algoritme vil forbedre ytelsen. Dersom dette er ønskelig, kan man utvide ESM med slik funksjonalitet senere.

5.4.7 Oppsummering og konklusjon

I dette avsnittet er det presentert to forskjellige angrepsmåter for allokering av jobber i en kjøreplan for en båndstasjon. Den første metoden er garantert å gi en optimal utnyttelse av ressursene, men er forholdsvis kompleks og gir lite fleksibilitet med hensyn til at ulike brukere har ulike bruksbehov. Den andre metoden bygger på at brukerne har forskjellige former for behov når de skal hente ut video fra bånd, og skiller derfor mellom dem. Ulempen er at det blir vanskeligere å finne ut om ressursene utnyttes maksimalt. Den siste metoden er forholdsvis lett å konstruere og implementere, og den er forholdsvis intuitiv. Derfor er det valgt å gå videre med denne i konstruksjonen.

5.5 Informasjon om system og estimering av verdier

ESM må ha en oppfatning om hvordan komponentene som den benytter seg av vil oppføre seg for å blant annet kunne beregne hvor lang tid en gitt forespørsel vil ta. Dette er nødvendig fordi ressursfordeleren må vite dette når den skal legge inn jobber i kjøreplanene. ESM bør blant annet kjenne til følgende verdier for en forespørsel:

- Overføringsrater for båndbiblioteket som skal benyttes.
- Aksesstid for arkivet som skal hentes (tiden det vil ta fra systemet bestemmer seg for å overføre en spesifikk video i båndbiblioteket, til de første dataene ankommer harddisk).
- Mengden data som skal hentes ut.
- Last på videotjenerens videopumper, som skal ta imot dataene.
- Prioriteten til brukeren som har sendt forespørselen.

5.5.1 Estimering av overføringsrate

Overføringsrater vil stort sett være omtrent lik fra gang til gang på en og samme båndstasjon, så dette er en parameter som kan settes statisk i en oppslagskatalog som beskriver båndstasjonene

etter en tids testing av båndstasjonen.[10]

5.5.2 Mengden av data som skal overføres

Datamengden kan finnes ved å slå opp i metadatabaser (se avsnitt 5.8). Det er imidlertid litt forskjellige vanskelighetsgrad på utregningene, avhengig av om brukeren har bedt om hele eller deler av en videostrøm:

- Dersom brukeren ønsker å hente inn en hel videostrøm fra bånd, kan ESM bare slå opp direkte i metadatabasen for arkiver og lese ut verdien dirkekte.
- Dersom brukeren kun ønsker visse mediasegmenter fra en videostrøm må ESM slå opp i metadatabasen for båndfiler og addere sammen lengden av alle filene som skal hentes inn.
- Dersom brukeren kun ønsker visse tidsintervaller av bestemte videostrømmer, må ESM beregne lengde ut fra f.eks. blokkadresser i indekser.

Uansett hva brukerne gis muligheten til å gjøre, så kan ESM finne den nøyaktige mengden data som skal overføres i det forespørselen ankommer.

5.5.3 Estimering av aksesstid

Det er verre å finne aksesstider, siden disse vil variere sterkt fra forespørsel til forespørsel (se avsnitt 4.1, side 30). Man kan tenke seg følgende løsninger på dette problemet:

1. Sette aksesstiden lik den høyeste aksesstiden som noensinne er målt under testing.
2. For hvert bånd (dvs. hver båndlengde) kan man måle aksesstider til hver eneste blokkadresse og lagre disse verdiene i en database. Denne databasen kan senere konsulteres når man skal ha tilgang til en blokk.
3. Utføre et detaljstudium av båndstasjonen for å så komme frem med en svært nøyaktig modell for hvordan den vil oppføre seg.

Alternativ 1 er svært sikkert, om enn noe brutalt. En serpentinstasjon vil ha svært varierende aksesstider, fra noen sekunder til flere minutter, noe som er tydelig vist av (Sætre og Sandstå 1997). Dersom man skulle sette aksesstid til å være f.eks. 2 minutter, betyr det at brukerne ikke har mulighet til å legge inn så mange jobber for kjøring i en gitt periode som de egentlig burde kunne gjøre.

Alternativ 2 krever endel arbeid i forbindelse med introduksjon av nye båndlengder og det krever dessuten plass på harddisk. Dersom man f.eks. har 300 000 innslag av 8 byte hver,[11] krever dette rundt 2.4MB lagerplass. Dette er ikke en avskrekkende størrelse. Beklageligvis kan informasjonen i en slik katalog være i uoverensstemmelse med de søketidene man opplever. Dette skyldes at når båndstasjonen oppdager ødelagte blokker, vil dette bli «reparert» ved at den lagrer informasjonen i en annen blokk.

Alternativ 3 krever at modellen som er utviklet er svært nøyaktig. Dette er imidlertid den mest elegante metoden, siden man slipper å lagre informasjon som egentlig ikke er påkrevet. I likhet med alternativ 2 vil imidlertid en slik løsning være utsatt for unøyaktigheter i forbindelse med feilkorrigering.

Det vil imidlertid gjerne være snakk om aksess til mer en ett sted på et bånd når en bruker skal hente ut informasjon, slik som vist i forrige avsnitt. Beregningen av aksesstider vil altså være avhengig av hva slags type forespørsel brukeren har gjort.

[10] Å kun basere seg på tall fra leverandøren er naivt.
[11] Blokkaddresse og antall sekunder, begge representert i 32 bits

Det er også en annen faktor som vil spille inn ved aksesstider, og det er hvor mange av forespørslene som går mot samme videobånd i båndarkivet. Siden tiden for å veksle to bånd i et arkiv kan være svært høy i forhold til aksess og overføringstid, virker det fornuftig at systemet bør ordne forespørslene slik at et minimalt antall vekslinger finner sted innen et tidsintervall. Forespørsler mot samme bånd kan i tillegg legges etterhverandre i en rekkefølge som fører til minst mulig akkumulert aksesstid, ved å bruke en algoritme lignende en av de som ble nevnt i avsnitt 3.5.1, side 26.

5.5.4 Informasjon om brukere

Elvira vil typisk betjene kunder befinner seg i forskjellige organisasjoner, og datasystemene de bruker er ikke under en felles administrasjon. Dette innebærer at man ikke kan gjøre seg avhengig av f.eks. UNIX' egne identifiserings- og adgangskontroll-mekanismer, siden man da måtte lagt inn alle brukerne av Elvira på det domenet der tjeneren skal kjøre.

Løsningen på dette problemet er at Elvira får sin egen database over brukere. Det som kan være aktuelt å lagre for hver enkelt bruker er:

- brukeridentifikator
- for- og etternavn
- prioritet
- e-postadresse
- passord
- prioritet
- avdeling
- organisasjon

Når f.eks. ressursfordeleren skal finne ut hva slags prioritet brukeren har, så slår den opp i databasen med brukeridentifikatoren som nøkkel.

5.5.5 Oppsummering og konklusjon

Overføringsrate og mengden data som skal overføres kan finnes nøyaktig, og med en gang en forespørsel ankommer ESM. Aksesstiden er det umulig å regne ut helt nøyaktig, men den kan enten overestimeres, måles på forhånd eller beregnes ut i fra en modell. De to siste metodene gir omtrentelige verdier, og det er derfor interessant å finne ut hvor nøyaktige de er. ESM bør ha en brukerdatabase for å kunne finne ut hva f.eks. slags prioritet en forespørsel kan få.

Valg av metode for estimering av aksesstid må sees i lys av at det også er andre faktorer som påvirker tiden det tar å laste inn data fra bånd til harddisk. Dersom systemet kun lagrer svært store datafiler, vil overføringsraten fullstendig overskygge viktigheten av aksesstiden. Dersom man har et manuelt arkiv, vil vekslingstidene for bånd antakeligvis være av større betydning enn aksesstiden, slik at man kan benytte høyeste målte aksesstid uten at effektiviteten av systemet blir merkbart dårligere.

5.6 Kontroll av harddiskbuffer

Dersom ikke data blir slettet fra harddiskbufferet med jevne mellomrom, vil det snart gå fullt. ESM må derfor ha funksjonalitet for å slette videomateriale som ikke trengs lenger. Dette er et velkjent

problem i forbindelse med virtuelt minne,[12] som er behandlet utførlig i litteraturen (Silberschatz og Galvin 1994, Wilkinson 1991). Forskjellen er at det her er snakk om langt større ventetider dersom man skifter ut noe som en bruker ønsker seg like etterpå.

5.6.1 Kjente algoritmer for utskifting

Her presenteres velkjente algoritmer for utskifting av data i et virtuelt minnesystem:

Tilfeldig

Tilfeldig utkasting av data kan faktisk være en optimal algoritme i et system der brukernes forespørsler ikke følger noe som helst mønster. Denne algoritmen er svært enkel å implementere, siden man bare behøver å foreta en trekning og deretter slette «taperen».

FIFO

FIFO-algoritmen er en kø-ordning. Denne algoritmen vil alltid fjerne de dataene som har ligget lengst i bufferet. Denne algoritmen antar at dataene vanligvis kun blir brukt en eneste gang, og at man deretter kan slette dem (etter en viss tid). Implementasjon av en FIFO er noe mer krevende enn for tilfeldig utkasting, og så sant antagelsene blir oppfylt, er den svært effektiv.

LRU

Den velkjente LRU-algoritmen (least recently used) består i å fjerne de dataene som ikke har blitt aksessert på lenge. LRU baserer seg på prinsippet om at materiale som akkurat har blitt brukt, snart bli brukt igjen. Implementering av en LRU-algoritme er noe mer arbeidskrevende enn for en FIFO, men det er slettes ikke en stor jobb. Når det gjelder bruk av systemressurser, er LRU noe mer krevende, siden den krever at innslag oppdateres hver gang de blir brukt. En FIFO-implementasjon krever bare én oppdatering, og dette skjer ved innhenting av en videofilm.

5.6.2 Modifiserte algoritmer

ESM har endel egenskaper som muliggjør visse modifiseringer av FIFO eller LRU slik at de blir mer effektive.

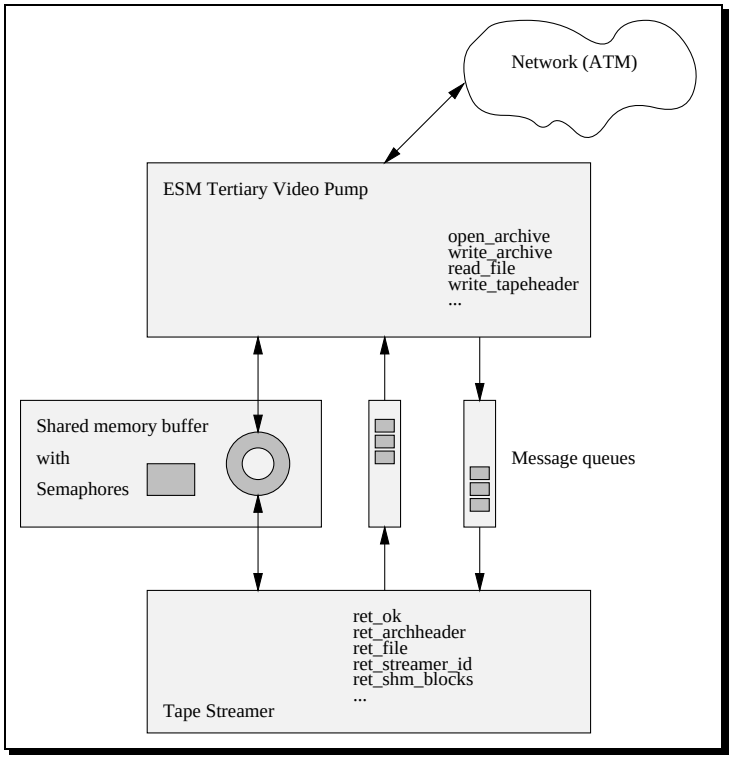
Foroverkobling

Problemet med utskiftingsalgoritmer er at så lenge det er umulig å se inn i fremtiden, er det umulig å oppnå en optimal utskiftingsalgoritme. Men er dette tilfelle med ESM? Nei, fordi ESM vil ha en ressursfordeler der forespørsler kan ligge i lang tid før de blir betjent. Det kan være at en bruker har forespurt en hel videostrøm, der deler av videostrømmen allerede ligger på harddisk. Det kan derfor være aktuelt å modifisere FIFO eller LRU med dette ekstra kravet: *såfremt ikke dataene er etterspurt i en forespørsel som ligger i ESMs køer.*

Låsing

Siden administratorene er mennesker, vil de som regel ha en bedre oppfattelse av miljøet rundt Elvira enn det som er mulig å legge inn i et dataprogram. Det vil derfor være aktuelt å tillate

[12] ESM er jo tross alt en slags minnehåndterer som bruker harddisk og bånd isteden for hovedminne og harddisk.



Figur 5.13: Grensesnitt mellom ETS og ESM

låsing av visse data i harddiskbufferet. Låsing kan gjøres med forskjellige grader av styrke, slik at ESM tar hensyn til dette dersom den *må* bryte noen låser. Modifiseringen av FIFO eller LRU blir da som følger: *såfremt dataene ikke er låst.*

5.6.3 Oppsummering og konklusjon

Tre velkjente algoritmer for utskifting av materiale i et buffer er presentert her. Det er valgt å gå videre med LRU fordi denne enkelt kan konverteres (eller «nedgraderes») til en av de to andre dersom man vil eksperimentere med forskjellige algoritmer i systemet. Det er presentert to modifikasjoner for FIFO og LRU som muligens kan øke effektiviteten, nemlig hensyn til bestillinger av materiale og mulighet for manuell låsing av visse videofilmer i harddiskbufferet.

5.7 Overføring av data fra bånd til harddiskbuffer

Selve overføringen av data fra bånd til harddiskbuffer er en av de mest sentrale problemområdene i ESM. Først ser vi på hvordan ESM skal kommunisere med ETS og med videopumpene. Deretter blir behovet for bufring av data underveis diskutert.

5.7.1 Kommunikasjon med Elvira Tape Server (ETS)

Kommunikasjonen med ETS er gjort svært enkelt gjennom klient-grensesnittet omtalt i avsnitt 5.1.4. Dette ble utviklet av Sætre (1997), i samarbeid med undertegnede og veileder.

Figur 5.13 viser hvordan grensesnittet mellom ETS og ESMs tertiære videopumpe vil se ut. Kommandoer sendes til ETS via en meldingskø (message queue) og svarene returneres via en annen

meldingskø. Når data skal overføres til eller fra ETS, brukes et ringbuffer i det delte minnebufferet (shared memory buffer). Det kalles et ringbuffer, fordi man har et begrenset minneområde og to pekere som peker inn i dette minneområdet. Den ene hodepekeren (head) refererer til begynnelsen av dataene, mens halepekeren (tail) peker til slutten av dataene som skal overføres. Semaforer brukes for å sikre at kun en prosess kan oppdatere pekerne av gangen. Følgende funksjoner vil finnes i klientgrensesnittet som ESM skal benytte:

- **GetBlock** – Returnerer en peker til en blokk i ringbufferet. Når man kaller GetBlock må man spesifisere om man skriver (head) eller leser (tail). Tilgangen til denne blokken blir låst for den andre prosessen.
- **ReleaseBlock** – Frigjør en blokk og brukes når en skrive eller leseoperasjon mot bufferet er utført.
- **AvailableBlocks** – Returnerer antall blokker i ringbufferet som ikke er opptatt.
- **StreamerID** – Forespørsel om hvilken ID båndstasjonen har.
- **WriteTapeHeader** – Initialisér et bånd ved å skrive en båndheader.
- **ReadTapeHeader** – Les båndheader fra båndet som står i stasjonen.
- **WriteArchive** – Skriv et arkiv til bånd. ETS må ha en arkivheader som sier hvor mye data som skal lagres, etc. Data blir deretter skrevet til bufferet via en peker returnert fra GetBlock.
- **OpenArchive** – Spesifisér et arkiv som skal åpnes for lesing. Brukeren angir hvilket arkiv som skal åpnes i arkivheaderen i argumentet til funksjonen.
- **ReadFile** – Les en båndfil fra et arkiv som er åpnet. Brukeren spesifiserer båndfilnummer.
- **ReadData** – Les en del av en båndfil fra et arkiv som er åpnet. Brukeren spesifiserer båndfilnummer, startblokk og lengde av intervall-lengde.
- **ReadPosition** – Returnerer blokknummeret stasjonen er posisjonert på.

Klientgrensesnittet oversetter så disse kommandoene til tekstbeskjeder, som sendes til ETS slik som vist i figur 5.13. Tjeneren utfører operasjonene i henhold til beskjedene den får, og oversetter returverdiene fra funksjonene den benytter til tekstbeskjeder som så sendes tilbake til klienten, dvs. ESMs tertiære videopumpe. Klientgrensesnittets funksjoner konverterer tekstbeskjedene og returnerer dem.

5.7.2 Kommunikasjon med videopumper og Command Server (CS)

Når ESM skal overføre data fra bånd til harddisk, kan den ikke skrive rett til Elviras filområde for harddisker. Man kan jo tenkte seg hva som skjedde dersom man plutselig tok all I/U-båndbredde på harddiskene mens Elvira var opptatt med å betjene klienter som må ha video overført med en bestemt rate. Resultatet ville i beste fall bli et hakkete bilde, i verste fall et stillbilde eller svart skjerm. Elvira2 vil ikke tillate en slik situasjon å oppstå i det hele tatt.

ESM må derfor samarbeide med Elviras kontrollsystem for å passe på at overføring av data til harddiskbufferet holder seg innenfor en grense som gjør at brukerne av Elvira ikke merker noe. Dette kan gjøres på forskjellige måter:

- Spørre CS om å stoppe videopumpene helt i en viss tid, slik at ESM kan skrive data direkte til diskområdene.
- Be CS om å reservere kapasitet på pumpene, og deretter overføre data fra ESM til pumpene, som da vil kjøre i revers-modus og skrive data fra nettverk til disk.

Dersom Elvira blir sterkt belastet med mange kunder, vil en mellomlagringsdisk, slik som beskrevet i avsnitt 5.7.3, være svært aktuelt. Man vil da ha mulighet for å vente med å overføre data til belastningen på harddiskbufferet går ned.

5.7.3 Bruk av buffere i dataoverføring

Her blir to former for bufring av datastrømmen på dens vei fra bånd til harddiskbufferet diskutert. Bufring gjøres for å sikre at båndstasjonen kan lese kontinuerlig, uten å måtte stoppe opp fordi systemet klarer å overføre dataene til harddiskene raskt nok.

Delt minnebuffer

Et delt minnebuffer mellom båndstasjon og ESM vil være fornuftig, siden man kan risikere at ESM ikke kan sende dataene umiddelbart videre til harddiskbufferet, enten fordi nettverket er overbelastet en kort stund, eller at man venter på at harddiskene blir mindre belastet, slik at de er klare til å ta imot data.

Bufferet i en typisk båndstasjon er akkurat stort nok til å holde en, eller i høyden noen få, blokker. Dersom båndstasjonen oppdager at bufferet ikke er lest, vil den stoppe, og ikke starte opp igjen før ESM leser bufferet. Dersom slike små forsinkelser som nevnt ovenfor skjer ofte, vil dette gå katastrofalt ut over overføringsraten, siden en båndstasjon bruker forholdsvis lang tid på å starte lesingen igjen (Silberschatz og Hillyer 1996*c*).

Mellomlagringsdisk

En mellomlagringsdisk[13] vil fungere som et temporært diskbuffer der ESM kan lagre data som leses inn fra båndstasjonen før de blir sendt videre til videopumpene. En mellomlagringsdisk vil typisk ha plass til mange hundre megabyte, helst flere gigabyte, med data før den blir full. Prinsippet er altså det samme som for et delt minnebuffer, men en mellomlagringsdisk har 2-3 størrelsesordner høyere kapasitet.

En slik disk vil kun betjenes av en enkelt prosess, nemlig ESM, for overføring av data. Dette sikrer at hele I/U-båndbredden til disken kan benyttes i overføring mellom bånd og disk. Deretter kan man flytte data fra denne disken til diskene som brukes av videopumpene når en eller flere av videopumpene har lav belastning.

Ulempen med mellomlagringsdisker er at kostnadene øker, siden den ikke kan brukes til noe annet fornuftig når det ikke er behov for bufring. Dersom den ble brukt til noe annet ville mye av poenget falle bort, siden dagens båndstasjoner kan levere flere MB per sekund, noe som kan være nok til å fylle mer eller mindre hele båndbredden til en vanlig harddisk. Fordelen er at kravene til størrelse på ringbufferet minskes, og man har en garanti om at systemet alltid vil være i stand til å ta imot data fra båndstasjonen i lengre tid.

Innføringen av en mellomlagringsdisk vil nok være mest nyttig dersom båndstasjoner eller andre tertiære lagerenheter med svært stor overføringsrate blir tatt i bruk, slik at et buffer i hovedminnet vil ta for stor plass. Det kan også være aktuelt dersom harddiskbufferet i Elvira er veldig opptatt med å betjene mange samtidige kunder, slik at adgangen til diskbufferet er stengt for ESM.

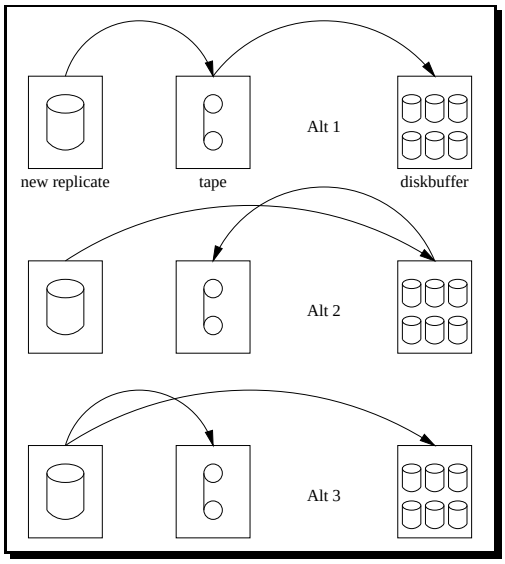
Dersom man antar at en mellomlagringsdisk burde kunne holde det en båndstasjon klarer å levere i løpet av 100 minutter, med kontinuerlig overføringsrate på 1.5MB/s. Disken må da ha en kapasitet på 9 GB. Dette er et verste-fall-scenario, siden effektiv overføringsrate fra en båndstasjon over så lang tid vil være langt lavere, på grunn av søking, bytting av bånd, osv.

5.7.4 Innlesing av nytt materiale i Elvira

Til nå er Elvira beskrevet som et system der det kun foregår leseoperasjoner. Det må selvsagt også finnes funksjonalitet for å legge inn nytt materiale.

[13] gjerne kalt «staging disk» i litteraturen

Det finnes tre mulige måter å introdusere nye videofilmer i Elvira/ESM på, som vist under og i figur 5.14:



Figur 5.14: Alternative vandringsveier ved introduksjon av videofilmer i Elvira/ESM

1. Lese den nye videofilmen til bånd fra en eller annen harddisk der man har formattert materialet slik at det kan lagres i ESM. Dersom brukerne har et umiddelbart behov for å se videoen rett etter at den er lagt inn er dette en dårlig løsning, siden brukerne må hente inn videostrømmen på vanlig måte, i konkurranse med alle andre forespørsler.
2. Skrive den nye videofilmen inn i diskbufferet ved hjelp av Elvira2s pumper (i revers-modus), og merke det slik at ESM må lagre det på bånd når det skal skiftes ut.
3. Ha mulighet for å gjøre begge deler i samme operasjon, dvs. at videodataene leses fra harddisken de ligger på og sendes til både Elvira2s videopumper og ESM tertiærpumpe. Disse to deloperasjonene må ikke nødvendigvis foregå parallelt, men resultatet av operasjonen vil være at videofilmen ligger både i harddiskbufferet og på bånd etterpå.

Uansett hvilken løsning som velges, må metadatabaser og ECM oppdateres av ESM. Dette er diskutert videre i avsnitt 5.8.6. Siden Elvira2s videopumper ennå ikke har funksjonalitet for lagre data, er alternativ 1 benyttet videre i oppgaven.

5.7.5 Lagringsstrukturer

I et båndbibliotek vil det være i hovedsak to måter å lagre informasjon på. Man kan ha et mediasegment lagret i sin helhet på et bånd, eller spre blokkene over flere bånd som så må leses samtidig. Dette er striping slik som diskutert i avsnitt 1.2, side 2.

Fordelene med det første alternativet er at man vil få en enklere konstruksjon. I et operativt arkiv vil det i tillegg være enklere å kontrollere hvor materialet ligger fysisk, slik at f.eks. sikkerhetskopiering av en videofilm blir en enklere operasjon. Fordelen med det andre alternativet er en jevnere belastning av alle stasjonene, slik som diskutert i avsnitt 1.2, men det er også store ulemper med striping, som diskutert samme sted.

5.7.6 Oppsummering og konklusjon

Ved overføring fra bånd til harddisk vil ESMs tertiære videopumpe måtte benytte seg av ETS, CS og videopumpene i Elvira. Den tertiære videopumpen må kjøre på samme vertsmaskin som ETS fordi grensesnittet er implementert med System V IPC. Overføringen av dataene til harddisk skjer via Elviras videopumper, og styres av CS.

Man kan ha både delte minnebuffer og mellomlagringsdisker for å få datastrømmene til å flyte jevnt. I et lite system med få båndstasjoner, slik som testversjonen av Elvira, burde det være unødvendig å ha en egen disk for bufring. Å ha et minnebuffer er det imidlertid mye å tjene på, siden man typisk vil ha kortvarige perioder der systemet venter på at disker skal bli klare, etc.

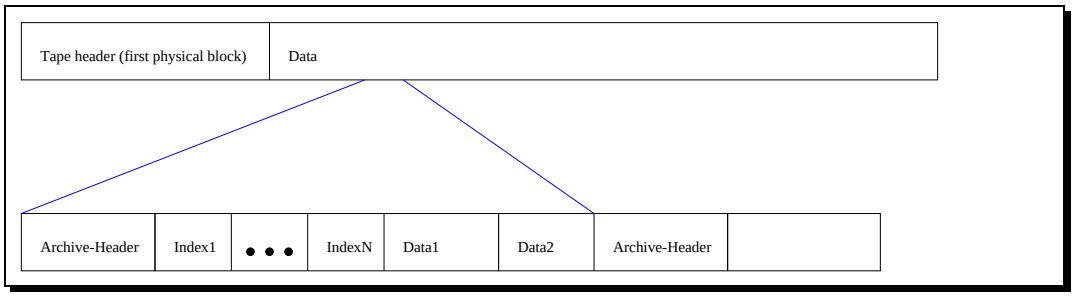
Når nye videofilmer skal leses inn i Elvira, blir de først lagret på bånd, og må deretter hentes ut på samme måte som filmer som allerede ligger inne. I en senere versjon av Elvira/ESM vil dette antageligvis bli endret.

Det er ikke aktuelt å bruke striping ved lagring på bånd. Dette ville ført til større kompleksitet i systemet, og i denne oppgaven er det for få tilgjengelige båndstasjoner til å få implementert noe slikt.

5.8 Metadatabaser og bufring av metadata

I Elvira er det allerede en modul for lagring av metadata, nemlig ECM (Sørensen og Sandstå 1996). Denne er imidlertid begrenset til å lagre informasjon om materialet som er av interesse for *alle* andre moduler. Dette avsnittet tar for seg ESMs behov for metadatabaser og et eventuelt behov for bufring av metadata som tar stor plass. Til slutt kommer en diskusjon om ESM og oppdatering av ECM.

5.8.1 Aksesser mot bånd



Figur 5.15: Skisse av båndformat

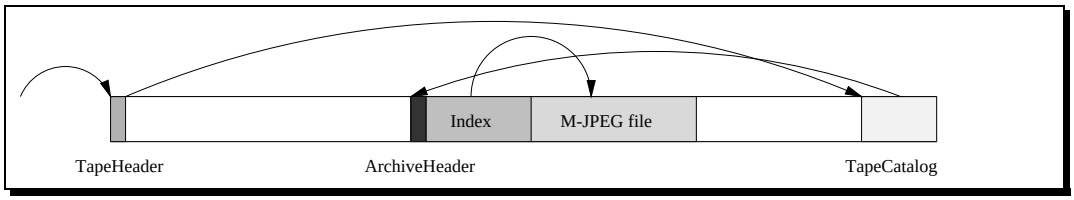
Figur 5.15 viser en sterkt forenklet modell av hvordan videodata er lagret på båndet. På begynnelsen av hvert bånd er en fysisk blokk allokert for å identifisere båndet (Sætre 1997). I tillegg har man en del informasjon som kan være nyttig i arkivsammenheng, som f.eks. om båndet er en kopi av et annet bånd og datoen båndet ble brukt første gang.

Det kan være mye å tjene på å minimalisere antall båndaksesser som kreves for hver henting av et mediasegment fra bånd, fordi en aksess[14] kan være svært tidskrevende, fra flere sekunder til i verste fall noen minutter. Dersom man ensidig skulle basere seg på den informasjonen som ligger på bånd for å finne en viss bildesekvens i et mediasegment, ville utføringen vært som følger:

[14] finne et arkiv og begynne overføringen av data

1. Dersom det står feil bånd i båndstasjonen, kast ut dette og plassér det på korrekt plass i arkivet.
2. Finn båndet som mediasegmentet befinner seg på. Opplysninger om hvilket bånd arkivet befinner seg *må* selvsagt befinne seg i en metadatabase på harddisk, ellers må *hele* båndbiblioteket søkes sekvensielt. Last båndet inn i stasjonen og gjør det klart for lesing.
3. Finn kataloginformasjonen for båndet (ligger på et fast sted) og finn blokkadresse for mediasegmentet). Dersom slik informasjon ikke finnes: les *hele* båndet for å finne arkivet som inneholder mediasegmentet.
4. Søk frem til og les arkivheaderen for å finne adresser til indeksene som beskriver innholdet av filmen.
5. Finn indeksen og bruk den til å søke frem til og lese blokkene som inneholder den ønskede lyd- eller bildesekvens.

I de følgende avsnittene evalueres forskjellige løsninger for å minske aksessene som er nevnt ovenfor. For å illustrere forbedringene, antas følgende: det skal hentes et mediasegment som kun inneholder én M-JPEG videofil, og en indeks der man kan finne blokknummer ut fra tidskode for filmen, slik som illustrert i figur 5.16.



Figur 5.16: Aksesser mot bånd uten metadatabaser og bufring av indekser

Dersom man må skifte bånd, burde man lese «tapeheader» også. Dette bør nok gjøres ved hver aksess fordi man vil forsikre seg om at man leser fra korrekt bånd før man setter i gang med tidkrevende søk etter video ute på båndet. Dette vil ikke koste stort, siden denne befinner seg helt på begynnelsen av båndet, og båndstasjonen bruker likevel tid for å posisjonere seg når båndet monteres.

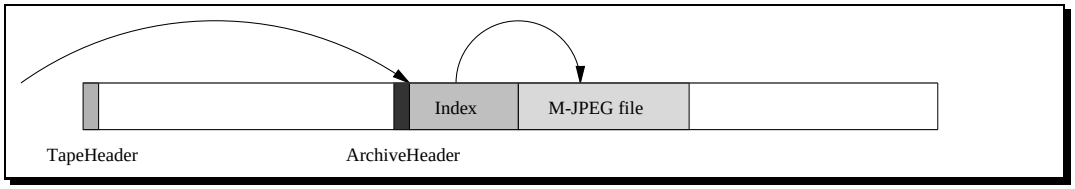
5.8.2 Database for bånd-identifikatorer

I punkt 2 ser man at det er behov for en database på harddisk som forteller oss i på hvilket bånd et gitt mediasegment befinner seg på. Å lete igjennom et arkiv på flere terabyte sekvensielt er et håpløst prosjekt. Det er derfor en nødvendighet å ha en metadatabase der man kan finne en bånd-identifikator ved å angi arkividentifikator.

5.8.3 Database for arkivheadere

Dersom man har lagret arkivheadere i en database, som nevnt i punkt 3, unngår systemet å 1) måtte søke gjennom hele båndet på leting etter rett mediasegment eller 2) slå opp i et eget katalogspor på båndet. Siden ESM vet både båndnummer og blokkadresse for hvor arkivet der mediasegmentet er lagret, kan man få båndstasjonen til å gå direkte til arkivet som inneholder mediasegmentet. Man må fortsatt søke seg frem til riktig indeksinnslag, og deretter søke seg frem til videodataene.

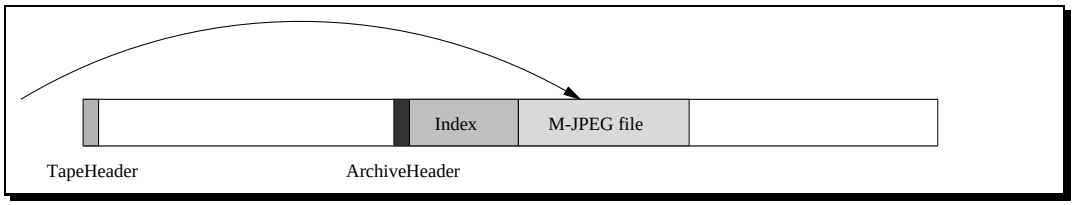
Nå er antallet aksesser mot båndet begrenset til 2, som vist i figur 5.17.



Figur 5.17: Aksesser mot bånd med metadatabaser, men uten bufring av indekser

5.8.4 Bufring av indeksfiler

Indeksfiler vil utgjøre en så stor mengde data, at det er urealistisk å lagre alle indeksfiler for alle mediasegmenter på disk, se C.1.2. Dersom man likevel hadde gjort det, ville man være i stand til å begrenset en aksess til en tilfeldig bildesekvens i et mediasegment til kun én aksess mot båndet, og resten mot disk, slik som vist i figur 5.18.



Figur 5.18: Aksesser mot bånd med metadatabaser og bufring av indekser

Imidlertid kan en god løsning være å bufre opp de mest brukte indeksene på et begrenset diskområde. På samme måte som man har et harddiskbuffer for mediadata, har man et harddiskbuffer for dataene som beskriver mediadataene. Problemstillingen angående utskifting av materiale fra dette bufferet vil stort sett være den samme som den som ble diskutert i avsnitt 5.6.

En av ulempene med lesing av deler av båndfiler er muligheten for at noen andre etterspør samme båndfil, men ønsker f.eks. å hente ut hele mediasegmentet. Enten må ESM laste inn hele mediasegmentet på nytt, ellers må det utvikles et forholdsvis komplekst system som kun leser de nødvendige delene av filene fra båndbiblioteket og så setter dem sammen til hele filer i harddiskbufferet. Dette vil sannsynligvis ikke utgjøre noe stort problem i en testimplementasjon av Elvira/ESM.

5.8.5 Hvor skal metainformasjonen ligge?

Som det går frem av diskusjonen ovenfor, er det mulig å lagre metainformasjonen slik:

- kun på disk.
- kun på bånd.
- både på disk og bånd.
- velge en av de tre ovenstående alternativene for forskjellige typer metadata.

Systemet vil spare tid på å begrense antall båndaksesser, mot å bruke større plass på harddisker til slik informasjon.[15] Her er en kort diskusjon om hva som bør gjøres i for hver type metadata, og en diskusjon om hva som skjer dersom databasen går tapt ved f.eks. et harddisk-krasj.

[15] Det er et vanlig at når man optimaliserer på hastighet så går det utover plassforbruk.

- En oppslagsdatabase for å få ut båndidentifikatorer når man har arkividentifikatorer er helt nødvendig. Dersom denne databasen ikke finnes, måtte man søkt gjennom hele båndbiblioteket hver gang man skulle ha tak i et mediasegment, og det ville tatt altfor lang tid. Dersom denne basen skulle gå tapt, ville det derfor være en katastrofe.
- En oppslagsdatabase for å få ut blokkadresser for arkiver når man har arkividentifikatorer er også svært nyttig. Det går også an å ha en katalog over arkiver lagret i et eget dataområde på selve båndene, men dette ville gått svært tregt. Dersom denne basen skulle gå tapt, er dette også rimelig katastrofalt, siden man ville måtte søke gjennom alle båndene for å genererer en ny indeks.
- En oppslagsdatabase for å få ut blokkadresser for båndfiler er like enkelt å implementere som i forrige punkt. Så sant ikke arkivet skal leses i sin helhet, spares man for en ekstra båndaksess. Det er ingen katastrofe om denne informasjonen skulle gå tapt, siden en ekstra båndaksess ikke tar allverdens tid, så basen kunne bli gjenoppbygd ved normal bruk.
- Indeksfiler ville tatt for mye plass på disk, så det eneste alternativet man har er å bufre dem. Dersom disse skulle gå tapt, vil det ikke være noen katastrofe, siden man bare må utføre en ekstra båndaksess.

De to første metadatabasene kan med letthet slås sammen til én, fordi det vil være én båndidentifikator og én blokkaddresse per arkiv. I en arkivheader legges det derfor inn ett innslag som sier hvilket bånd arkivet befinner seg på og ett innslag som sier hvor på båndet arkivet befinner seg.

5.8.6 Oppdatering av ECM ved utskifting og innhenting

I ECM er det en kolonne i mediasegment-tabellen som sier om det befinner seg i harddiskbufferet eller på bånd og et innslag for hver fil som befinner seg i harddiskbufferet i diskfil-tabellen.

Utskiftingsmodulen i ESM må oppdatere ECM hver gang data skal slettes fra bufferet, slik at informasjonen i mediasegment-tabellen er konsistent med hva som virkelig finnes i bufferet (se avsnitt 5.1.3, side 47). Dette gjøres ved at utskiftingsmodulen benytter seg av ECMs klientgrensesnitt og kaller de nødvendige funksjonene der.

Likeledes må ESMs innhentingsmodul legge inn informasjon når data har blitt lest inn fra bånd og overført til harddiskbufferet. Her dukker det opp et behov for nok en metadatabase, nemlig en database som lagrer informasjon om status for hvert bånd, slik at systemet vet hvor mange ledige blokker det er, hvilken blokk som ble skrevet til sist, og så videre. Uten denne informasjonen, risikerer man å forsøke å skrive til et bånd som vil bli fullt under operasjonen, og dermed må den utføres omigjen på et nytt bånd.

5.8.7 Sikkerhet

Dersom den disklagrede metainformasjonen skulle gå tapt, vil det være en svært stor oppgave å skulle gå igjennom alle båndene med video for å rekonstruere denne informasjonen. I et arkiv med f.eks. 10000 bånd og en gjennomsnittlig prosesseringstid[16] på f.eks. 4 minutter (som er svært optimistisk), så vil det ta hele 666 timer å rekonstruere databasen. Dette innebærer at metainformasjonen må beskyttes godt mot ødeleggelse, og backup til bånd (eller andre harddisker) bør gjøres jevnlig.

5.8.8 Oppsummering og konklusjon

Det vil være fornuftig å ha metadatabaser for:

[16] dersom man skal samle inn tapeheadere

- Arkivheadere
- Båndfilheadere
- Status for enkeltbånd

I tillegg bør det finnes et system for bufring av indeksfiler, slik at det utføres et minimalt antall båndaksesser for hver forespørsel om innhenting av videofilm fra båndbiblioteket. Det må tas backup av metainformasjonen jevnlig, siden det vil ta lang tid å bygge dem opp igjen ved å gå igjennom båndarkivet fra start til slutt.

5.9 Avhengighet av lagringsformater

Konklusjonen i avsnitt 3.3 var at formater for lagring av media ganske sikkert vil endre seg over tid, og derfor bør ESM i så stor grad som mulig bør gjøre seg uavhengig av disse. For å oppnå uavhengighet kan følgende gjøres:

- Behandle mediadata som rådata (bytestreams) uten noen form for spesialbehandling så høyt opp i systemet som mulig. Dette sikrer at lavere nivåer av ESM (og ETS) slipper å oppdateres bare fordi et nytt format blir introdusert.
- All spesialbehandling av mediadata på grunnlag av format bør gjøres av moduler som kan «plugges» inn i Elvira. Dette skjer imidlertid ikke uten kostnader. Det kreves større omtanke under konstruksjon og implementasjon enn om man hadde ignorert dette problemet.

5.10 Oppsummering

ESM må samarbeide med andre moduler i Elvira, slik som Command Server som tar seg av kontroll av videoleveranser, videopumper som leverer data fra harddisk til klienter via nettverk, Elvira Catalog Manager som håndterer metainformasjon og Elvira Tape Server, som gir ESM adgang til bufret overføring av data til og fra båndstasjon.

Grensesnittet mot sluttbrukerne bør være informativt, slik at de kan ta beslutninger om bestilling av video på best mulig grunnlag. Det er aktuelt å lage brukergrensesnitt som kan aksesseres på WWW. Sluttbrukerne må blant annet spesifisere ønsket videofilm, kvalitet og lengde når bestilling av video gjøres. Det er aktuelt å la kun to prosesser i Elvira være i direkte kontakt med sluttbrukerne, nemlig Command Server og en metadatatjener, siden dette forenkler adgangskontroll, m.m. Brukere av WebSTAR eller VideoSTAR kan også komme i kontakt med Elvira/ESM dersom Elvira benyttes som videotjener for disse. Det bør være støtte for å kunne underrette brukere via f.eks. E-post når det skjer noe med forespørselen deres. Sluttbrukerne har typisk behov for enkle, grafiske brukergrensesnitt, mens administratorer ofte foretrekker kommandolinjebaserte.

ESM vil fungere som en tjener for Command Server, videopumper og administrator-applikasjoner, mens den vil være en klient av Elvira Catalog Manager, Command Server og Elvira Tape Server. Kommunikasjon mellom prosesser vil gjøres ved hjelp av System V IPC når de befinner seg på samme maskin, og via socket-grensesnittet mot TCP/IP når de befinner seg på forskjellige maskiner. Kommunikasjonsprotokoller vil være enkle.

På grunn av brukernes forskjellige behov i forbindelse med bestilling av video, virker løsningen med forskjellige forespørselskøer fornuftig. Vanlige brukere kan velge mellom rene bestillingsjobber (B-jobber), interaktive jobber (I-jobber) og uprioriterte jobber (U-jobber). I tillegg finnes det mulighet for å legge inn jobber som reserverer systemet i nødsituasjoner og for vedlikehold. Det må velges en politikk for ressursfordeling som utnytter systemet best mulig og samtidig tilfredsstiller et størst mulig antall brukere.

ESM må estimere endel verdier når forespørsler ankommer systemet. Disse verdiene skal brukes for å legge inn jobber på rett sted i kjøreplanen, slik at politikken for ressursfordeling holder. Blant annet må overføringsrate, mengden av data som skal overføres, aksesstid mot båndbibliotek og informasjon om brukerne hentes inn av systemet. Endel av disse verdiene kan bare bestemmes omtrentelig, og det er derfor viktig å ta høyde for disse, slik at systemet kan holde hva det lover.

Det må slettes data fra harddiskbufferet hver gang nytt materiale hentes inn, ellers vil det gå fullt. Til dette trengs det en algoritme for utskiftning. Godheten av utskiftingsalgoritmen er viktig fordi systemet ellers risikerer å kaste ut materiale som snart må brukes igjen.

Overføringen av data fra bånd til harddiskbuffer er avhengig av interaksjon med Elvira Tape Server, Command Server og videopumper. I denne testimplementasjonen av ESM vil det bli brukt et delt minnebuffer ved overføringer mellom ESM og ETS og bruk av sockets mellom ESM og videopumpene. Overføringsprosessen vil være sterkt avhengig av hva slags organisering det på dataene i båndbiblioteket. Det velges en løsning uten bruk av striping.

Bruken av metadatabaser på harddisk gjør at antallet aksesser mot båndet der videofilmen er lagret kan minskes betraktelig. Det er også aktuelt å bufre opp indekser av videofilmer som gjør systemet i stand til å lese inn kun de tidsintervallene som er interessante for brukeren. Dette vil redusere behovet for båndbredde mot båndbiblioteket. Metadatabasene må oppdateres hver gang data legges inn i Elvira/ESM og hver gang videofilmer overføres fra bånd til harddiskbuffer eller slettes fra harddiskbuffer.

Til slutt bør ESM gjøre seg så uavhengig av formater for lagring av data som mulig. Modulær oppbygging og bruk av såkalte innstikk-moduler kan gjøre ESM helt uavhengig av formater.

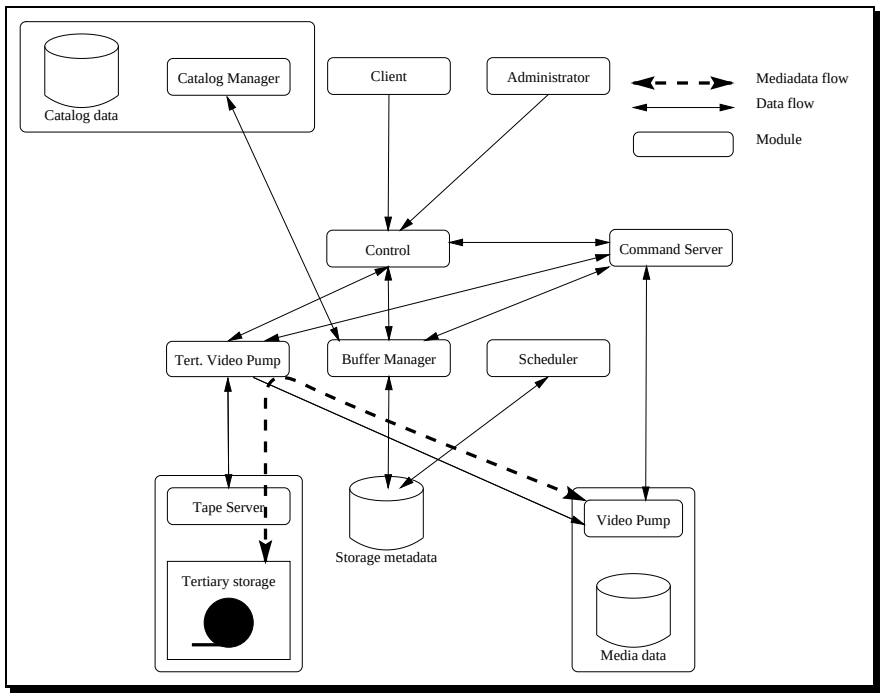
Kapittel 6

Konstruksjon

Dette kapittelet er inneholder overordnet konstruksjon for ESM. Den detaljerte konstruksjonen gikk i ett med implementasjonen, og er derfor beskrevet i kapittel 7, og i tillegg D. Visse detaljer er imidlertid også tatt med her, for å klargjøre hva som skjer i de forskjellige delmodulene. Kapittelet er delt inn slik at hvert avsnitt inneholder en beskrivelse av en delmodul. Først kommer imidlertid en oversikt over ESM.

Merk at i selve implementasjonen vil alle klasser få prefikset «ESM_» for å sikre at klassene er unike innen Elvira-kildekoden. I konstruksjonen er disse prefiksene utelatt for ESM-klasser, mens eksterne klasser har fullt navn.

6.1 Oversikt over ESM



Figur 6.1: Overordnet arkitektur for Storage Manager

Ut i fra de viktigste problemområdene i forrige kapittel, er det naturlig å dele opp ESM i moduler

som løser hvert sitt problemområde:

- Control – kontrollerer de andre modulene og er «hjernen» i ESM. Den er også sentral i kommunikasjon med andre deler av Elvira.
- Tertiary Video Pump – en modul som henter data ved hjelp av Elvira Tape Server (ETS) og sender dem videre til en av Elviras videopumper (som går i «revers»).
- Scheduler – en modul som ordner køen av forespørsler for innhenting av mediadata fra bånd til diskbuffer.
- Buffer Manager – en modul som organiserer plassforbruket på harddiskene, slik at de ikke fylles opp med materiale som ikke blir brukt. Den er også ansvarlig for å frigi plass i harddiskbufferet når ESM skal hente videofilmer fra båndbiblioteket.

I figur 6.1 ser man hvordan de forskjellige modulene vil samarbeide innad i ESM og mot andre moduler.

6.1.1 Applikasjonsgrensesnitt

Applikasjonsgrensesnittet er todelt. Funksjoner som skal kalles fra andre moduler via nettverket ligger i en egen klient-bibliotek, som gjør de andre modulene i stand til å kommunisere med ESM via nettverket. Typiske administrasjonsfunksjoner ligger derimot i programmer som må kjøres på vertsmaskinen til ESM. En oversikt over funksjoner er vist i tabell 6.1.

Navn	Beskrivelse
Klient	
Acquire	Be ESM om å hente video fra bånd til harddiskbuffer.
Reply	Se hva ESM svarte på forespørselenn.
Confirm	Bekreft at man vil benytte seg av tilbudet ESM gav.
Cancel	Avlys kjøringen av en jobb man har lagt inn.
Mark Used	Marker et mediasegment som brukt (brukes av bufferhåndtereren).
Lock	Marker at et mediasegment ikke skal skiftes ut av Buffer Manager.
Unlock	Frigjør et mediasegment slik at det kan skiftes ut av Buffer Manager.
Status	Returnerer status for ESM.
Import av data i ESM	
Set Archive Parameters	Legg inn nødvendig metainformasjonen for et arkiv.
Add File To Archive	Registrer en datafil for senere innlegging i et båndarkiv.
Store Archive	Be ESM om å lagre et båndarkiv på disk.
Register Tape	Registrer et nytt bånd i metadatabasen for bånd.
Ressursfordeler-administrasjon	
Initialize scheduler	Slett alle jobb-køer.

Tabell 6.1: API-funksjoner for ESM

6.1.2 Bruk av eksisterende kode

I flere av avsnittene nevnes klassen TCP_Socket. Dette er en klasse laget for Elvira av Langørgen (1994). TCP_Socket tilbyr et objektorientert grensesnitt mot ANSI C socket-biblioteket, som følger med blant annet SunOS5. Sockets er kort beskrevet i avsnitt 5.3.2, side 54.

Klassen ET_Client er utviklet av Sætre (1997) og tilbyr grensesnittet som ble beskrevet i avsnitt 5.1.4, side 48.

6.2 Ressursfordeler

Løsningen med heterogene jobbforespørsler fra avsnitt 5.4.2, side 59, ble valgt fordi den er rimelig enkel å konstruere og fordi den antageligvis vil støtte brukernes krav på en bedre måte.

6.2.1 Overordnede regler for planlegging av forskjellige jobber

Det må finnes regler for hvordan jobber skal køes i systemet. Her følger den politikken som er valgt for ESM:

- Alle E-jobber blir øyeblikkelig prosessert.
- B-jobber skal kun brukes for jobber som trengs et godt stykke frem i tid, og man har derfor en sperre på hvor nære et tidspunkt man kan være for å sende en forespørsel av denne typen. En B-jobb kan kun skyve ut en annen B-jobb dersom:
  1. den andre B-jobben fortsatt kan fortsatt utføres innen sin tidsfrist.
  2. den nye B-jobben ikke fører til at belastningen av båndstasjonen kommer over en viss prosentandel over et bestemt tidsrom.
- Dersom brukeren ber om en I-jobb så vil systemet svare med hvor lang tid det tar før jobben blir utført (den kan selvsagt ikke forutse nødsituasjoner). Brukeren må så svare bekreftende eller avkreftende på om hun fortsatt vil at materialet skal hentes inn. I-jobber kan man ikke bestille til et bestemt tidspunkt. I-jobber kan kun legges inn i nåtidig tidsintervall. Brukerne kan legge inn slike jobber så lenge systemet har kapasitet til å ta seg av dem innenfor tidsintervallet.
- U-jobber benytter ledig kapasitet i systemet og henter inn data i disse dødperiodene. Systemet vil kjøre U-jobber før man henter inn batch-jobber fra neste tidsintervall, slik at man får en rimelig responstid på disse jobbene også. Det er mulig at denne regelen burde tilsidesettes dersom det er svært mange batch-jobber i neste tidsintervall. Tidspunktet som blir angitt av brukeren er kun for å hindre data i å bli tatt inn for tidlig, slik at bufferet fylles av unødvendig informasjon.

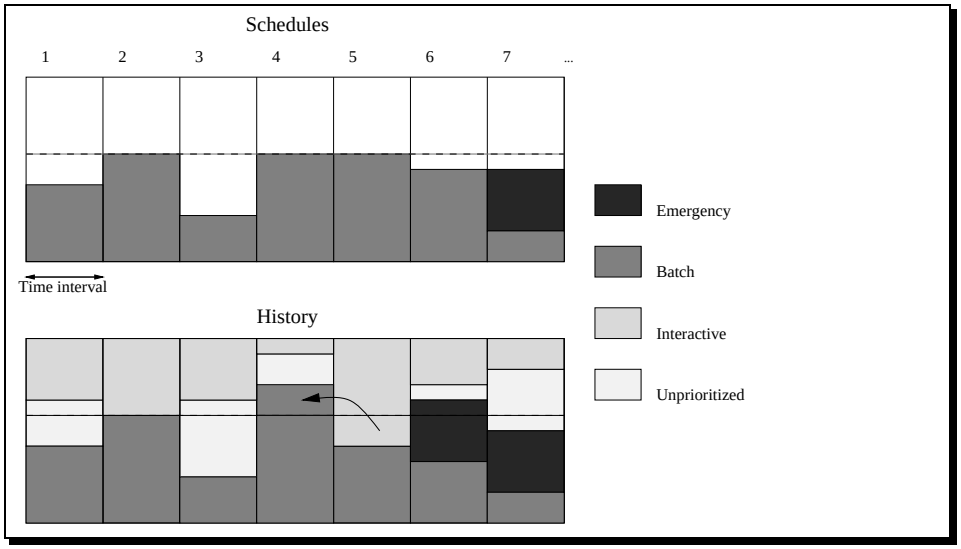
Et resultat av en slik politikk, kan f.eks. for en enkelt båndstasjon være slik som vist i Figur 6.2. Øverst ser man hvordan kjøreplanen så ut rett før tidsintervall 1 ble nåtid. Nederst ser man resultatet av kjøring av alle tidsintervallene, og er altså en historisk beskrivelse av hendelsesforløpet i ressursfordeleren.

Øverst ser vi at det er lagt inn B-jobber i hvert eneste fremtidige tidsintervall, og at det er lagt inn en nødjobb i intervall 7. Nederst ser vi resultatet av kjøringen. Belastningen på systemet har vært stort, og hvert eneste tidsintervall frem til intervall 5 har blitt utnyttet fullt ut. Da intervall 4 ble utført, var det ledig kapasitet til overs, så en B-jobb ble flyttet fra neste intervall. Dette førte så til at det kunne utføres flere interaktive jobber i intervall 5. En nødjobb ankom intervall 6 under utføring. Man kan også observere hvordan U-jobber hele tiden fyller inn hullene i kjøreplanen, og de er dermed med på å øke effektiviteten av systemet.

6.2.2 Objektbeskrivelse

Ressursfordeleren vil bestå av en rekke klasser, som vist under og i figur 6.3.

- Brukerjobb (UserJob) – en jobbforespørsel der brukeren har angitt type (B, I, U eller E), tidsfrist, brukeridentifikator og eventuelt hvor mye materiale som ønskes (valg av mediasegmenter).



Figur 6.2: Tenkt eksempel for bruk av en båndstasjon over tid

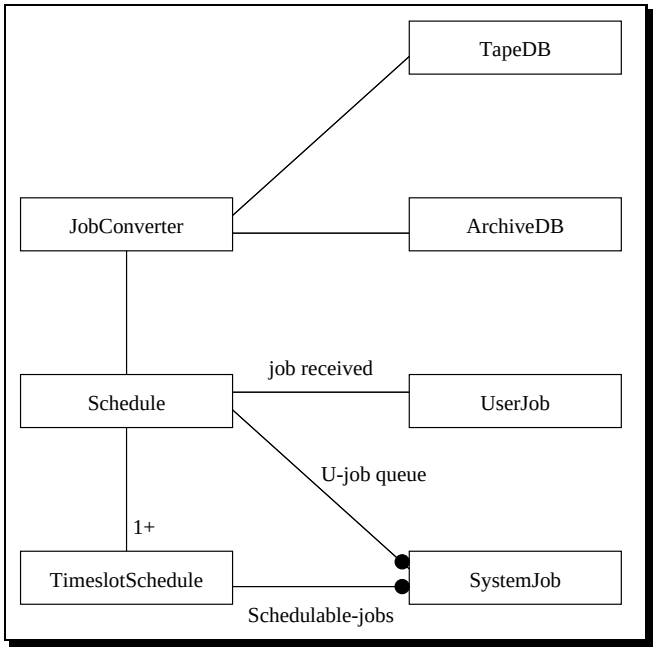
- Systemjobb (SysJob) – en jobbforespørsel der systemet har lagt til all den informasjon som er nødvendig for å kunne foreta en fornuftig tildeling av en plass i en kjøreplan for den.
- Kjøreplan (Schedule) – inneholder en liste av kjøreplaner for hvert tidsintervall, samt en kø av U-jobber.
- Kjøreplan for tidsintervall (TimeslotSchedule) – kan deles inn i tre typer:
  - Fremtidige – inneholder kun en oversikt over hvilke B-jobber som skal kjøres, ingen spikret kjøreplan. Eventuelle E-jobber er derimot registrert med start og slutt-tid.
  - Nåtidige – inneholder en spikret kjøreplan for B- og eventuelle E-jobber. I-jobber legges inn etterhvert som de ankommer.
  - Historiske – lagrer hvordan kjøreplanen for tidsintervallet virkelig ble gjennomført, for alle jobber.
- Jobbkonverterer (JobConverter) – bruker de dataene som brukeren har oppgitt i en UserJob og henter inn den informasjonen som mangler for å kunne lage en SysJob. Denne informasjonen hentes fra metadatabaser beskrevet i avsnitt 6.6

6.2.3 Funksjonell beskrivelse

Her presenteres algoritmen for ressursfordeleren. Hvert underavsnitt inneholder en beskrivelse av hva en liten del av programmet vil gjøre. Disse små delene vil stort sett finnes igjen som metoder i objektene i implementasjonen. Beskrivelsen begynner på høyt nivå og referer til metoder som er beskrevet i med flere detaljer og mindre abstraksjon.

6.2.3.1 Utfør valgt forespørsel

1. Dersom brukeren spør om innlegging av jobb, utfør 6.2.3.2.
2. Dersom brukeren vil kansellere en jobb, utfør 6.2.3.7
3. Dersom tertiærpumpen er ledig, vil systemet be ressursfordleren om å få en ny jobb å utføre. Den nye jobben hentes fra kjøreplanen. Dersom det ikke er noen jobb å utføre i nåværende tidsintervall, utfør 6.2.3.6.



Figur 6.3: Objektmodell for ressursfordeler

6.2.3.2 Innlegging av jobb

1. Ta i mot UserJob.
2. Utfør 6.2.3.3. Dersom dette feilet, gi feilmelding til bruker.
3. Utfør 6.2.3.4. Dersom dette feilet, gi feilmelding til bruker.
4. Gi tilbakemelding til bruker og vent på bekreftelse.
5. Når bruker gir bekreftelse, oppdater kjøreplanene.

6.2.3.3 Konverter UserJob til SysJob

1. Ta imot UserJob.
2. Sjekk at bruker finnes og at vedkommende har lov til å benytte seg av tjenesten hun ber om.
3. Sjekk at brukeren har bedt om en tjeneste som er lovlig ut fra systemtilstanden. Man har for eksempel ikke lov til å legge inn immediatejobber for annet enn nåværende tidsintervall, eller B-jobber for annet en fremtidige tidsintervall.
4. Bruk parametrene for å slå opp i metadatabaser der man finner ytterligere informasjon som trenges i ressursfordelingen. Dette kan være informasjon om hvilket bånd og hvor på båndet filmen befinner seg på, et estimat for aksesstid på denne filmen, osv.
5. Generer SysJobs som brukes av ressursfordeleren. Denne typen forespørsler inneholder mye mer informasjon enn UserJobs, og inneholder blant annet estimater på hvor lang tid en gitt operasjon vil ta.

6.2.3.4 Legg inn SysJob i Schedule

1. Ta i mot SysJob.
2. Finn ut hvilket tidsintervall jobben hører hjemme i. U-jobber legges i en egen FIFO-kø.

3. Utfør 6.2.3.5 med SysJob som parameter.
4. Oppdater status dersom forrige operasjon var vellykket.

6.2.3.5 Legg inn SysJob i TimeSlotSchedule

1. Plassér alle E-jobber der de hører hjemme. Kansellér alle jobber som kommer i konflikt med disse, se 6.2.3.7.
2. Plassér alle A-jobber der de hører hjemme, dersom de ikke kolliderer med E-jobber.
3. Dersom det er en B-jobb, sjekk om den prosentvise grensen for B-jobber i et intervall er oversteget. Dersom det er ledig kapasitet og intervallet har fremtidsstatus, registrer B-jobben. (Nøyaktig plassering gjøres først når tidsintervallet får nåtidsstatus).
4. I-jobber som ankommer et tidsintervall med B-jobber, kan bli utført umiddelbart (de må riktignok ordnes seg i mellom) så lenge det er nok tid til å utføre B-jobbene før tidsintervallets utløp. Dette er «utsettelsespolitikken» nevnt i avsnitt 5.4.2, side 59.
5. Dersom det ankommer en U-jobb, og systemet er helt ledig i dette tidsintervallet, utfør den med det samme, så sant brukeren ikke har spesifisert at det er lenge til vedkommende trenger denne filmen. Dersom systemet er opptatt, legges U-jobben i den egne køen for slike jobber i Schedule.

6.2.3.6 Flytt B-jobb

1. Sjekk at jobben som bes flyttet er en B-jobb, det er kun denne typen jobber som kan flyttes nærmere nåtid.
2. Sjekk at B-jobben som skal flyttes ikke ligger lenger fram i tid enn en grense spesifisert av administrator. Denne grensen er satt for å hindre at diskbufferet skal inneholde for mange filmer som venter på å bli sett.
3. Sjekk at B-jobben som skal utføres ligger i neste tidsintervall eller høyere.
4. Dersom alt har gått greit til nå, legg inn B-jobb for kjøring i nåværende tidsintervall, og slett innslaget i det tidsintervallet B-jobben ble hentet fra.
5. Send beskjed til bruker om at jobben vil utføres raskere enn planlagt.

6.2.3.7 Kansellér jobb

1. Sjekk at brukeren som ber om kansellering er autorisert til å gjøre det.
   - Vanlige brukere har kun lov til å kansellere sine egne jobber.
   - Administratorer har lov til å slette hvilken som helst forespørsel.
   - Alle typer jobber kan kanselleres implisitt ved innlegging av E-jobber.
2. Slett jobben fra tidsintervallet der den hører hjemme.

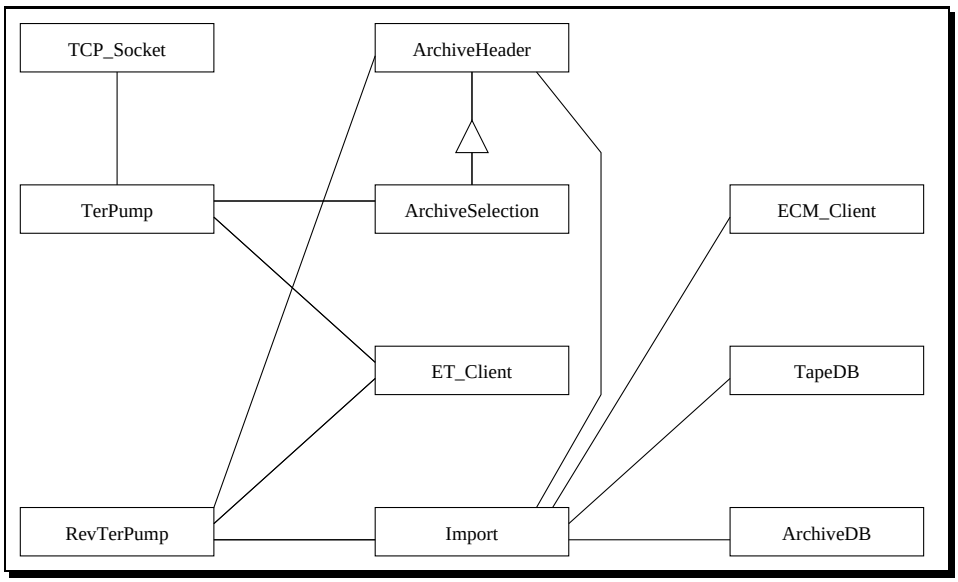
6.3 Tertiær videopumpe

Den tertiære videopumpen vil ha to modi, en for å overføre data fra bånd til Elviras videopumper (normal modus) og en for å overføre data fra et vilkårlig diskområdet og inn på bånd (revers modus). Siden disse oppgavene vil bli svært forskjellige, hører de hjemme i hver sine klasser.

6.3.1 Objektbeskrivelse

Den tertiære videopumpen vil bestå av objektene vist under. Relasjonene mellom dem er vist i figur 6.4.

- Normal tertiær videopumpe (TerPump) – som tar i mot et forespørsel om et sett av filer å hente ut av et arkiv (ArchiveSelection), og bruker ET_Client for å lese disse. Dataene blir overført til Elviras videopumper via en TCP_Socket.
- Revers tertiær videopumpe (RevTerPump) – som tar i mot en ArchiveHeader med informasjon om et nytt arkiv som skal lagres ved hjelp av ET_Client.
- Import – som generer et båndarkiv ved hjelp av rådata, informasjon gitt av administrator. Import oppdaterer TapeDB og ArchiveDB og ECM (via ECM_Client) for å gjøre den nye videostrømmen tilgjengelig for brukerne. Deretter kan dataene overføres til RevTerPump.
- ET_Client brukes for å kommunisere med ETS.
- TCP_Socket er et grensesnitt mot nettverket.
- ArchiveSelection er en ArchiveHeader med mulighet for å velge ut individuelle båndfiler.
- ArchiveHeader, TapeDB og ArchiveDB er beskrevet i avsnitt 6.6.



Figur 6.4: Objektmodell for tertiær videopumpe og relaterte klasser

6.3.2 Funksjonell beskrivelse

Her presenteres de forskjellige metodene i den tertiære videopumpen (TerPump) og den reverse tertiære videopumpen (RevTerPump). Handlingsgangen i disse metodene er ikke beskrevet i detalj, siden de utfører rimelig enkle oppgaver, i motsetning til hos f.eks. ressursfordeleren. Bare de viktigste metodene er tatt med.

6.3.2.1 Tertiær videopumpe

open
→ archiveheader
← success
Denne metoden åpner et såkalt *image* som er en stor diskfil som inneholder alle datablokkene etter arkivheader, nøyaktig i den rekkefølgen som de vil bli lagret på bånd, byte for byte.

init_com
→ host, port
← success
Be tertiær videopumpe om å kople seg opp til en av Elviras videopumper, spesifisert ved navnet på vertsmaskinen og portnummeret som denne videopumpen vil ta i mot data på.

start_transmisssion
→ void
← success
Start selve overføringen av data.

6.3.2.2 Revers tertiær videopumpe

open_image
→ filename, archiveheader
← success
Denne metoden åpner et såkalt *image* som er en fil som er klargjort for lagring på bånd (den inneholder alle datablokkene etter arkivheader), nøyaktig i den rekkefølgen som vil brukes på bånd, byte for byte.

store_image
→ void
← success
Starter selve overføringen til båndet.

6.4 Plasshåndterer for harddiskbuffer

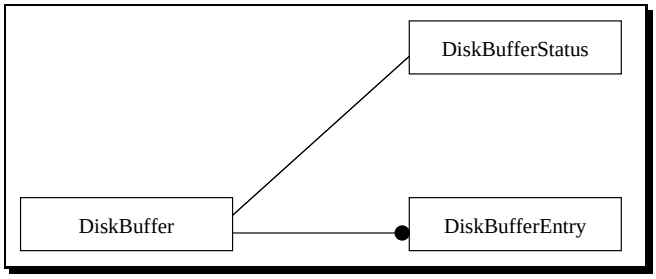
Det er ikke nok å ha en effektiv algoritme for å avgjøre i hvilken rekkefølge data skal hentes inn fra tertiærminne til harddisk. Det må også slettes like store mengder data (over tid) på harddiskene, slik at ikke bufferet blir fylt. Plasshåndtereren tar seg av dette ved å lagre metainformasjon som beskriver innslagene i bufferet.

6.4.1 Objektbeskrivelse

Her er objektbeskrivelsen for delmodulen Buffer Manager. Figur 6.5 viser hvordan klassene er relatert til hverandre.

- DiskBuffer – består av en DiskBufferStatus og et DiskBufferEntry for hvert mediasegment som befinner seg i bufferet.
- DiskBufferStatus – inneholder informasjon om total kapasitet for bufferet, ledig kapasitet og mengden av låste områder.
- DiskBufferEntry – inneholder informasjon om et mediasegment som befinner seg i bufferet: identifikator for mediasegment, tidsstempling for når mediasegmentet ble benyttet sist og

størrelsen av mediasegmentet.



Figur 6.5: Objektmodell for bufferhåndterer

6.4.2 Funksjonell beskrivelse

Metodene som følger er grensesnittet mot plasshåndtereren. Det er ikke mening at brukerne skal ha direkte adgang til dette grensesnittet, siden adgangskontroll ikke er implementert her. Dette vil kontroll-modulen ta seg av. Det er også viktig å få med seg at plasshåndtereren ikke utfører de faktiske slette-operasjonene. Den tilbyr kun informasjon som gjør det mulig for kontrollmodulen å be Elviras videopumper om å slette de rette innslagene i bufferet.

acquire
→ id, size
← success
Allokér plass for et diskinnslag som kan identifiseres med **id** og har størrelse **size**. Dersom dette går bra, returneres en sannhetsverdi for **success**. Alle diskinnslag som er markert for sletting kan listes ut ved hjelp av **victim**-funksjonene nedenfor.

release
→ id
← success
Frigjør plassen som et diskinnslag som identifiseres med **id** bruker.

use
→ id
← success
Marker diskinnslaget identifisert med **id** som benyttet, slik at det får en ny tidsstempling. Dersom dette ikke gjøres, vil LRU-rutinen reduseres til ren FIFO.

lock/unlock
→ id
← success
Disse metodene brukes for å kunne låse en eller flere diskinnslag, slik at de ikke blir slettet. Dette kan f.eks. være nødvendig for mediasegmenter som kun befinner seg på harddisk og ikke på bånd.

victim
→ void
← DiskBufferEntry
Det returnerte **DiskBufferEntry** identifiserer et offer som må slettes før man kan lagre mediasegmentet som man ba om plass for (med acquire).

firstvictim/nextvictim
→ void
← success
Disse metodene brukes for å iterere gjennom alle ofre etter et kall til **acquire**.

init
→ size
← success
Initialiserer metainformasjonen for et diskbuffer av størrelse **size**.

6.4.2.1 Alloker plass til et innslag

- Dersom det er ledig kapasitet i diskbufferet, bruk denne plassen, og marker den som brukt.
- Ellers må vi finne kandidater for sletting:
  1. Finn innslaget med laveste verdi i tidsstemplingsfeltet (dette er LRU).
  2. Sjekk om innslaget *kan* slettes (om det er låst, etc.)
  3. Marker innslaget som klart for sletting og oppdater den ledige kapasiteten i diskbufferet.

6.4.3 Bemerkninger

Bufferhåndtereren bør kun operere med lagringsstrukturer som er persistente, dvs. at de ligger på disk. Det ville ikke være bra om hele diskbufferet måtte slettes, bare fordi strømmen gikk, eller det skjedde en programfeil.

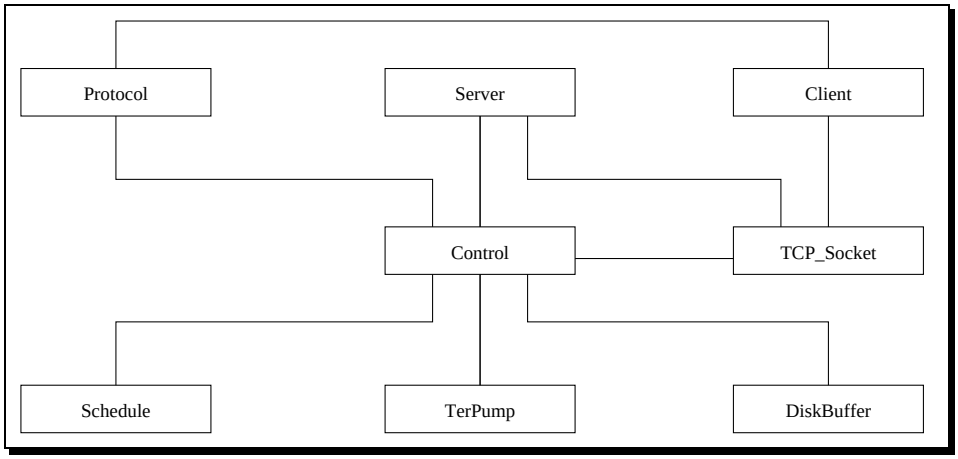
6.5 Kontroll og kommunikasjon

Modulen for kommunikasjon mot ESM vil ligne svært på den som ble laget for ECM. Kontrollmodulen er ESMs hjerne, og vil styre de tre modulene som utfører selve arbeidet. Dessverre viste det seg at det ikke ble tid til å implementere alle modulene som ESM skal bestå av, og det ble derfor besluttet å prioritere konstruksjon og implementasjon av ressursfordeler, bufferhåndterer, tertiær videopumpe og metadatabaser. Konstruksjonen beskrevet i dette avsnittet er derfor noe skisseaktig og ikke særlig detaljert.

6.5.1 Objektbeskrivelse

Kontroll og kommunikasjonsmodulen vil bestå av klassene beskrevet her og i figur 6.6.

- Control – er hjernen i ESM, og har forgreninger til alle de andre modulene. Bruker Protocol og Server for kommunikasjon, og Schedule, TerPump og DiskBuffer for å utføre tjenester.
- Protokoll-håndterer (Protocol) – Inneholder metoder for parsing og generering av beskjeder mellom Client og Server.
- Klientgrensesnitt (Client) – Inneholder metoder som applikasjoner kan benytte seg av for å kommunisere med ESM. Benytter seg av TCP_Socket for kommunikasjon og Protocol for tolking av beskjeder.
- Tjener (Server) – inneholder funksjonalitet for å få ESM til å fungere som en tjener for klienter. Benytter seg også av TCP_Socket og Protocol.
- Schedule, TerPump og DiskBuffer – beskrevet i andre avsnitt.



Figur 6.6: Objektmodell for kontroll og kommunikasjon

6.5.2 Tjener

Tjeneren vil være en svært enkel såkalt *seriell forbindelsesorientert tjener* (Robbins og Robbins 1996). Den er *seriell* fordi den kun betjener en forespørsel av gangen. Alle andre forespørsler blir køet og må vente til den aktive er ferdig. Den er *forbindelsesorientert*[1] fordi kommunikasjonen mellom klient og tjener kan bestå av flere beskjeder (forespørsler og svar) som går mellom dem. Grunnen til at denne løsningen er valgt, er at de fleste operasjoner vil ta svært kort tid, og at brukerne må få mulighet til å bekrefte svar som ESM gir i forbindelse med ressursfordeling. Store dataoverføringer vil imidlertid være et unntak fra reglen om serialitet, se avsnitt 6.5.5.

1. Tjeneren lager en socket som bindes til en port ved hjelp av metoder i TCP_Socket. Kun en prosess på en gitt maskin kan binde en socket til en bestemt port.
2. Tjeneren venter på innkommende beskjeder ved å kalle Accept() i TCP_Socket. Dette fører til at hele prosessen *sover* til en beskjed ankommer, og CPU-en blir dermed ikke unødvendig belastet.
3. Når en klient kopler seg til porten går tjeneren ut av sovemodus og det opprettes en ny socket (et nytt TCP_Socket-objekt) som skal brukes i kommunikasjonen. Denne sendes videre til Control.

6.5.3 Kontroll

1. Beskjeden leses ut fra kommunikasjons-socketen og dekodes ved å kalle parsefunksjonen i Protocol.
2. Avhengig av hva slags forespørsel som kom inn, kalles metoder i Scheduler, DiskBuffer eller TerPump.
3. Dersom beskjeden innleder en forespørsel som krever flere beskjeder, går Control inn i en tilstandsmaskin for behandlingen av denne forespørselen. Kommunikasjonen foregår ved hjelp av kommunikasjons-socketen og Protocol.
4. Kommunikasjons-socketen stenges og Control returnerer for å la Server vente på en ny beskjed.

[1] i motsetning til forbindelsesløsorienterte tjenere, der hver forespørsel inneholder all nødvendig informasjon for å utføre forespørselen, og tjeneren sender kun ett svar tilbake for hver klient som kopler seg til den.

6.5.4 Protokoll

Protokollen er en datastruktur som entydig representerer all den informasjon som trengs for at kontrollflyt mellom klient og tjener skal fungere. Det er valgt en tre-struktur av objekter som kan generere en tegnstrøm (stream) hos avsender. Denne strømmen brukes så for å bygge opp et identisk tre på mottakersiden. Strømmen sendes til rotnoden i treet som tolker om det er en forespørsel eller et svar. Følgende skjer:

1. Hver klasse som er en rotnode i treet kan kun parse informasjon som går direkte på datamedlemmene i klassen.
2. En rotnode har en eller flere pekere til «barneobjekter» som instansieres avhengig av hva slags informasjon rotnoden leste ut av strømmen.
3. Rotnoden sender strømmen videre til barnet som ble instansiert, som så vil begynne på punkt 1.
4. Dette vil skje helt til vi når en løvnode.

Vi har dermed at parsetre-objektene (som utgjør hele protokoll-modulen) faktisk er i stand til å parse seg selv. Inngangspunktet er rotnoden. Dersom man sender en tegnstrøm til rotnodens parsemetode, vil hele treet bygges av seg selv. Dersom man har bygd et tre, kaller man genereringsmetoden i rotnoden for å få en tegnstrøm.

Å implementere en protokoll-modul er enkelt, men krever nøyaktig arbeid for å unngå feil i kommunikasjonen, som kan være vanskelig å avluse. Et utkast til en protokoll-spesifikasjon er gitt i avsnitt D.1.1, side 129.

6.5.5 Prosess-struktur

Som nevnt i avsnitt 6.5.2 vil hele ESM stort sett kjøre som en enkelt prosess, som kun tar i mot en forespørsel av gangen. Det er likevel endel operasjoner som *må* foregå i parallell, nemlig dataoverføring mellom tertiær videopumpe og Elviras videopumper (diskvideopumper). Disse kan ta flere minutter, og slike ventetider ville gjort ESM nærmest ubrukelig.

6.6 Kataloger for metainformasjon

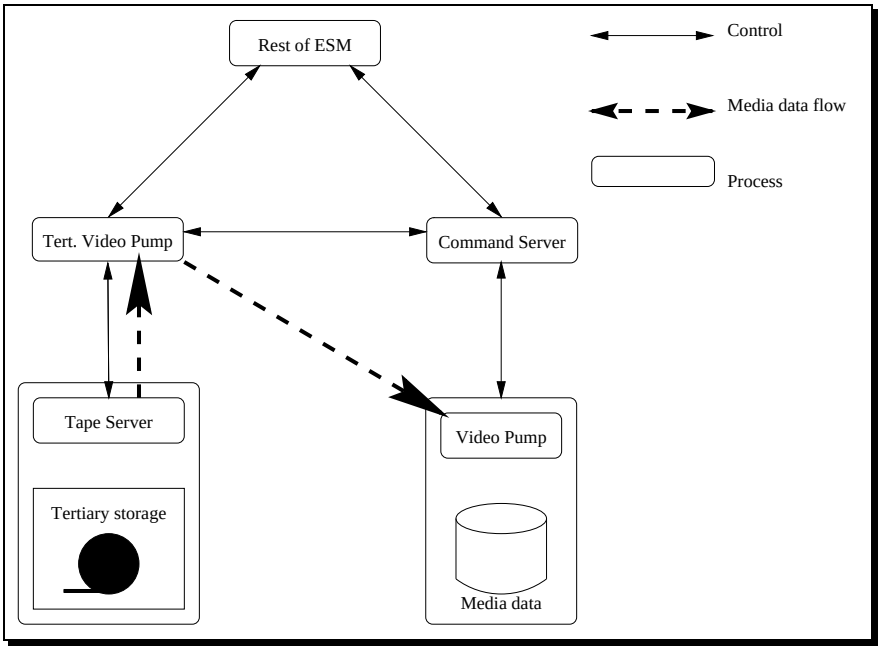
Det må lages en katalog som inneholder informasjon om datainnholdet i arkivet. Denne informasjonen brukes for å:

- Vite hvilket bånd man skal hente fra biblioteket når brukeren ber om et mediasegment som ikke finnes på disk.
- Vite hvor på båndet man skal finne filmen. Det er selvsagt mulig å søke seg fram på båndet, men det ville vært ineffektivt.

Man må også ha en oversikt over parametre for de båndstasjonene/bibliotekene som ESM benytter seg av.

For hver båndstasjon bør følgende være lagret:

- Maskin-ID for maskinen som båndstasjonen er koplet til
- Adresse for maskinen
- Hastighet
- Blokkstørrelse



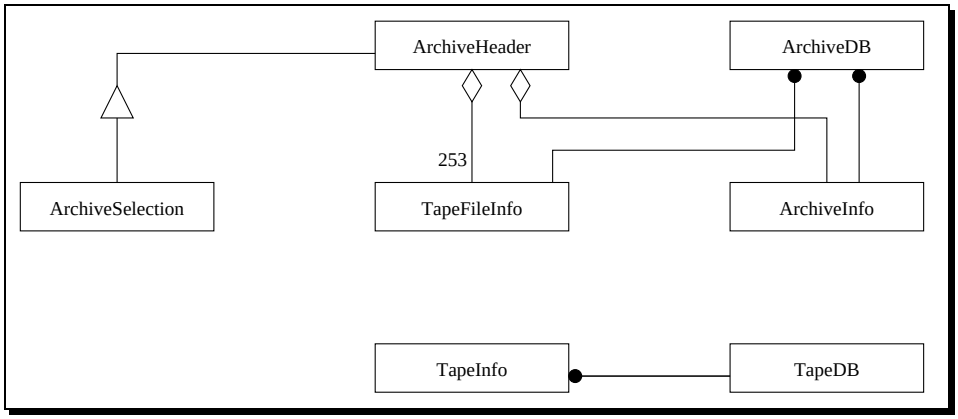
Figur 6.7: Prosesser-struktur under overføring av mediadata

6.6.1 Relasjoner mellom ESM og ECM

Det vi kalte for *videostrømmer* i ECM er det samme som *arkiver* på bånd. Likeledes er *lagrede mediasegmenter* i ECM representert som *filer* på bånd. I figur 6.9 vises Elviras metadatabase-konstruksjon som et ER-diagram. Dette er en logisk modell som er implementert gjennom to delmoduler, nemlig ESM og ECM. Skillet mellom dem er vist som stiplede linjer. Disse to er i fellesskap ansvarlige for konsistens.

6.6.2 Objektbeskrivelse

Metadatabasene til ESM vil i hovedsak bestå av klassene vist under. De vil imidlertid også benytte seg av klasser for persistent lagring av data, noe som vil bli diskutert i neste kapittel. Relasjonene mellom klassene er vist i figur 6.8.



Figur 6.8: Objektmodell for kontroll og kommunikasjon

- ArchiveDB – inneholder en eller flere ArchiveInfo og TapeFileInfo-objekter. Klassen tilbyr lagring av ett ArchiveInfo og en eller flere TapeFileInfo-objekter for hver arkiv-identifikator.
- ArchiveHeader – inneholder informasjon om et helt båndarkiv. Objekter av klassen ArchiveHeader er sammensatt av ett ArchiveInfo-objekt og ett eller flere TapeFile-objekter. Maksimalt antall TapeFileInfo-objekter er satt til 253 i testimplementasjonen av ESM, fordi man da får ArchiveHeader-objekter som er nøyaktig 32kB store.
- ArchiveSelection – er en spesiell form for ArchiveHeader, der det i tillegg er mulig å merke av TapeFileInfo-objekter som skal leses fra bånd.
- TapeFileInfo – inneholder beskrivelse av en båndfil. Se tabell 6.3.
- ArchiveInfo – inneholder beskrivelse av et arkiv. Se tabell 6.2.
- TapeDB – inneholder en eller flere TapeInfo-objekter. Hvert lagret TapeInfo-objekt kan hentes ut igjen med en bånd-identifikator.
- TapeInfo – inneholder status for et bånd, se tabell 6.4.

6.6.3 Database for arkivinnslag

Databasen for båndarkiv er en svært viktig ressurs for ESM. Denne databasen er ment å fungere som en indeks for dataene som er lagret på bånd. Dersom denne databasen forsvinner, må man bruke store ressurser på å lese igjennom alle båndene for å bygge opp informasjonen igjen. Det er derfor viktig at denne databasen holdes konsistent, og at man tar sikkerhetskopier av den (f.eks. til bånd) med jevne mellomrom.

ArchiveInfo		
field identifier	type	description etc.
archive_id	integer	unique key
archive_time	date	
archive_name	string	
header_address	integer	
header_blocks	integer	
no_files	integer	
tape_no	integer	(liste her!)

Tabell 6.2: Tabelldefinisjon for arkivinformasjon

I figur 6.9 er databasebeskrivelsen for arkivinnslagene vist. Egentlig er dette en litt kunstig oppdeling. I virkeligheten vil man ha en ArchiveHeader som består av en ArchiveEntry, og N TapeFiles. Lagringen i flere tabeller er kun gjort for å spare (tildels mye) plass på disk. I systemet vil man se på en stor ArchiveHeader på f.eks. 32kB (et ArchiveEntry og 253 TapeFiles).

TapeFile		
field identifier	type	description etc.
archive_id	integer	foreign key
file_no	integer	partial key
address	integer	
type	string	
size	integer	

Tabell 6.3: Tabelldefinisjon av fil på bånd

ArchiveHeader er en (eller flere) 32kB struct med informasjon om blant annet navn på arkivet, se tabell 6.2. En ArchiveHeader inneholder 0 til 253 TapeFile-structer. Hver TapeFile inneholder en peker til en blokken der dataene i blokken starter på båndet. Dersom man trenger mer enn 253 TapeFile-structer, så vil det automatisk legges inn flere ArchiveHeader-blokker i begynnelsen av arkivet, som beskrevet av Sætre og Sandstå (1997).

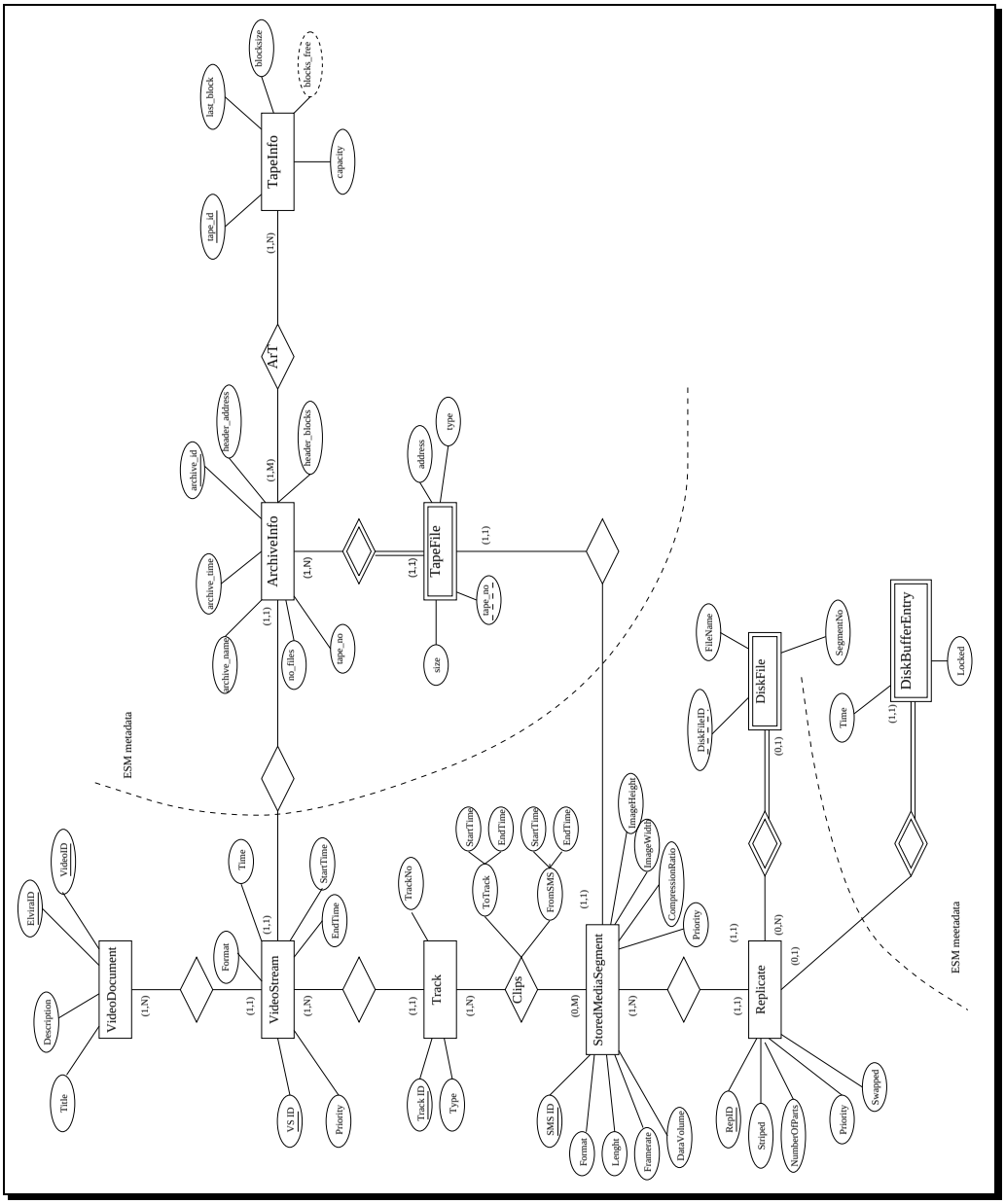
I tabell 6.3 er databasebeskrivelsen for en TapeFile vist. TapeFile representerer både datafiler og indeksfiler. En indeksfil vil typisk ha en tabell med tidskoder og tilsvarende offset i bytes inn i datafilen. Hvordan slike indekser blir generert, er utenfor rekkevidde av denne oppgaven, men det er ikke vanskelig å lage slike for f.eks. MPEG-1 videodata. Det finnes identifikatorer i datastrømmen som gjør at man kan telle seg igjennom en fil og generere en indeksfil svært enkelt. Uansett hva innholdet er.

6.6.4 Database for informasjon om bånd

Som nevnt i avsnitt 5.8, er det behov for katalog med status for hvert bånd som er registrert i Elvira/ESM. TapeInfo, vist i tabell 6.4, inneholder denne statusinformasjonen.

TapeInfo		
field identifier	type	description etc.
tape_id	integer	unique key
last_block	integer	last block written to
capacity	integer	capacity (in blocks) for tape
blocks_free	integer	number of available blocks on tape
block_size	integer	size of each block on tape

Tabell 6.4: Tabelldefinisjon for båndinformasjon



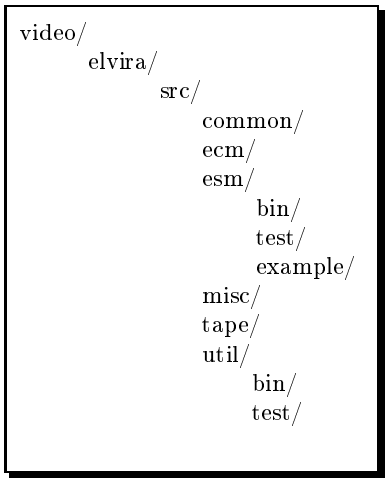
Figur 6.9: Databaseskjema for metadata i Elvira

Kapittel 7

Implementasjon

Dette kapittelet tar for seg status for implementasjon for de modulene som er beskrevet i konstruksjonen. Detaljert systemdokumentasjon finnes i tillegg D. Generiske klassebiblioteker som er utviklet av undertegnede under diplomarbeidet er også beskrevet her. Eksempler på kjøring av testprogrammer og resultater fra dette er tatt med til slutt.

Kildekoden er lagret i Elviras eget kildekodehierarki. I figur 7.1 er dette hierarkiet vist. Kun katalogene som blir brukt i forbindelse med denne oppgaven er vist.



Figur 7.1: Kildekodehierarki for ESM

7.1 Ressursfordeler

En god del av det som er beskrevet i konstruksjonen (avsnitt 6.2, side 79) er implementert. Det henvises til tillegg D for detaljer. Arbeidet ble ikke dessverre ikke helt ferdigstilt, men mye av «skjelettet» er på plass. Det finnes derfor ingen utskrifter av testkjøringer av denne modulen.

7.2 Tertiær videopumpe

Det aller meste av det som er beskrevet i konstruksjonen (avsnitt 6.3, side 82) er implementert og testet. Selve dataoverføringen er fullt funksjonell og kan taes i bruk i et ferdig ESM. Det er laget

en rekke programmer for å teste modulen, og resten av dette avsnittet beskriver disse.

7.2.1 Testprogram: timport

SYNOPSIS

```
timport.exe <archive_id> <numberoftimes> <imagename> <datafile> <dowritetape>

argument:        type:

archive_id       long
numberoftimes    int (<253)
imagename        char * (name of file where temporary image shall be stored)
datafile         char * (name of file to be imported)
writetotape      char (y/n) (write to tape, or just pretend)
```

ttimport er et program for å lagre mediadata på bånd. Programmet er ment for testing, og endel verdier er hardkodet inn i programmet, slik at det må endres før det kan inngå i en sluttversjon av ESM.

Programmet begynner med å gå gjennom alle filene som skal være med i arkivet og bruker metadatabasene for å generere de verdiene som skal til for å (1) få lagret arkivet på rett plass på båndet, (2) få korrekt informasjon i arkivheader og (3) få metadatabaser med konsistent innhold.

7.2.2 Testprogram: tvideopump

SYNOPSIS

```
  tvideopump.exe
```

tvideopump er laget fordi verken Elvira eller Elvira2 har videopumper som kan kjøres i revers (dvs. ta imot data). Dessverre viste det seg at TCP_Socket-klassen, som programmet baserer seg på, har en svakhet i en av metodene sine. Det var egentlig planlagt å utføre målinger med tmeasure der data blir sendt videre over nettverket til **tvideopump**, men dette måtte avlyses, på grunn av tidsmangel.

7.3 Plasshåndterer

Plasshåndtereren, slik den er beskrevet i avsnitt 6.4 er stort sett ferdig implementert. Funksjonalitet for å låse innslag i diskbufferet gjenstår. Testing har ikke blitt utført på denne koden.

7.4 Metadata

Dette avsnittet beskriver det som ble implementert av funksjonalitet for behandling av metadata. Alt som ble beskrevet i konstruksjonen (avsnitt 6.6, side 88) er implementert og testet. Det er laget flere testprogrammer som viser at implementasjonen fungerer tilfredsstillende, og en beskrivelse av hver av dem følger, med resultater fra kjøring som er kommentert.

7.4.1 Testprogram: ttapeadmin

SYNOPSIS

```
Usage: tapeadmin.exe <capacity> <blocksize> <writetotape>

argument:        type:

capacity         long
blocksize        long
writetotape      char (y/n) (write to tape, or just pretend)
```

ttapeadmin er et program for å registrere nye bånd i Elvira/ESM. Administrator angir kapasitet og blokkstørrelse for båndet som skal introduseres, og **ttapeadmin** vil deretter registrere disse dataene i TapeDB (se avsnitt 6.6.4, side 91), og skrive en TapeHeader-blokk til begynnelsen av båndet ved hjelp av ETS. Båndet er nå klart til å taes i bruk.

7.4.2 Testprogram: tmetadb

SYNOPSIS

```
tmetadb.exe l|t|f

option:    result:
t          list tapeheaders
a          list archiveheaders
f          list tapefiles
```

tmetadb er et program for å liste ut innholdet av det som ligger på bånd. Den bruker ESMs metadatabaser og lister ut innholdet formattert i kolonner, og fungerer i det hele tatt som ESMs svar på UNIX-kommandoen **ls(1)**.

7.5 Kontroll og kommunikasjon

Det viste seg at tiden ikke strakk til for å bli helt ferdig med implementasjonen, og derfor ble kontroll og kommunikasjons-modulene ofret. Årsaken var at det vil bli relativt enkelt å implementere dem, og et lignende arbeid har blitt gjort av undertegnede tidligere i forbindelse med ECM (Sørensen og Sandstå 1996). De andre modulene ble derfor prioritert.

7.6 Programmer for måling

Det ble utviklet flere programmer i C++ og Perl for måling og evaluering av den tertiære videopumpen. Disse programmene er presentert i dette avsnittet.

7.6.1 Måleprogram: tmeasure

SYNOPSIS

```
tmeasure.exe <numberofarchives>

argument:          type:
numberofarchives   long (iterates archives with id 1 .. numberofarchives)
```

tmeasure er et testprogram skrevet i C++ for å utføre tidsmålinger på lesing av data fra bånd. Den benytter en instrumentert versjon av klassen TerPump for å måle aksesstid og overføringstid for hver aksess til et arkiv, og skriver målingene til en loggfil som senere skal plottes (se kapittel 8).

7.6.2 Måleprogram: measure.pl

measure.pl er et program skrevet i Perl for å automatisere målinger av data.

7.6.3 Måleprogram: variance.pl

variance.pl er et program for gjentatte målinger ved hjelp av measure.pl. Hensikten er å gjøre målinger som er grunnlaget for å finne gjennomsnitts- og ekstremalverdier, samt standardavvik, for aksesstider.

7.6.4 Statistiske utregninger: statistics.pl

statistics.pl er et program som tar resultatene fra variance.pl og regner ut gjennomsnitts- og ekstremalverdier, standardavvik og 95% konfidensintervall ved 8 målinger. Eksempel på bruk finnes i neste kapittel.

7.7 Nyttige generelle klasser

For å lette arbeidet med implementeringen av ESM er det behov for endel generelle nytte-klasser som tilbyr tjenester som ikke finnes i normale ikke-kommersielle klassebiblioteker. Disse klassene er generelle, fordi de ikke nødvendigvis må benyttes i ESM, men kan benyttes av programmer i hele Elvira-systemet. De fleste klassene er faktisk så generelle at de kan benyttes i et hvilket som helst C++program som skal kjøre under UNIX.

Disse klassene har blitt implementert direkte når det har vært behov for dem, uten noen forutgående konstruksjon (unntaket er brukerdatabasen, som er laget spesielt for Elvira).

Når man skal benytte seg av de generelle klassene (som befinner seg i util-katalogen i Elvira) er fremgangsmåten slik (util-katalogen må angis, slik at G++ klarer å finne biblioteket):

SYNOPSIS

```
g++  [flag ... ] file ...  -lUtil [ library ... ]
```

7.7.1 Objektorientert grensesnitt mot GDBM

The GNU database manager, er et *svært enkelt* databasesystem. Det er faktisk så enkelt at det er misvisende å snakke om «system» i det hele tatt. Uansett: GDBM implementerer en randomisert filstruktur med bruk av hashingsalgoritmer (Nelson og Gaumond 1995). Det betyr at man vanligvis gjør svært få diskaksesser per oppslag (Bratbergsengen 1996).

GDBM er implementert i C. For å forenkle bruken av GDBM er det utviklet et C++ grensesnitt som gjør at det lett kan brukes av objekter som vil lagre andre objekter, så lenge disse andre objektene ikke inneholder pekere.

Grensesnittet består av fire klasser, der vi har arving som følger: RGDBM ← GDBM ← KCGDBM ← CCGDBM. RGDBM tilbyr kun lese-operasjoner (read-only) mot en database og GDBM tilbyr et grensesnitt som inneholder alle funksjonene som vi finner i C-versjonen av GDBM. KCGDBM

og CCGDBM er spesialklasser som tilbyr ekstra funksjonalitet som går på sammenligning av henholdsvis nøkler (key comparing) og innhold (content comparing). GDBM-grensesnittet er utførlig beskrevet i tillegg D.

7.7.2 Tidskoder

Tidskoder, eller TC-er (timecodes) som de vanligvis kalles i videoproduksjonsmiljøene her i Norge, er en oppdeling av tid i følgende format: `HH:MM:SS:FF`, der HH står for timer, MM for minutter, SS for sekunder og FF for rammenummer (framenumber). Klassen TimeCode støtter de aritmetiske operasjonene + og - på tidskoder. Hver gang man skal instansiere objekter av denne klassen, må man spesifisere hvor stor rammefrekvensen skal være.

7.7.3 Tidsangivelser

Tidsangivelser er alltid en kilde til problemer i samfunnet generelt og i UNIX spesielt. For det første så er grensesnittet mot de funksjonene som befinner seg i UNIX' bibliotek for tidsbehandling, <**time.h**>, mildest talt forferdelig. Funksjonene for å lage strenger produserer resultater som er lite hensiktsmessige («`Tue Feb 18 19:20:48 MET 1997`»). ISO8601 er en standard for hvordan strenger for tid skal formatteres, og man får isteden en angivelse i formatet «`1997-02-18 19:20:48-01:00`», som er mye mer entydig og mindre avhengig av at forskjellige parametre er konfigurert korrekt på maskinen programmet kjører på.

Klassene Local_Time og UTC_Time takler begge lesing og skriving av strenger som følger ISO 8601-standarden[1]. Den eneste forskjellen mellom de to klassene er at utskrift av tidsstrenger er av formen «`YYYY-MM-DD HH:MM:SS±HH:MM`» i Local_Time og «`YYYY-MM-DD HH:MM:SSZ`» i UTC_Time. Objekter av begge klasser kan gjensidig tilordnes hverandre.

7.7.4 Brukerdatabase

UserDB		
field identifier	type	description etc.
user_id	integer	unique key
user_name	string	alternate unique key
e-mail	string	user's e-mail address
priority	integer	priority of user
password	string	user's password encrypted
first_name	string	given name
second_name	string	family name

Tabell 7.1: Tabelldefinisjon av brukerdatabase

Brukerdatabasen vil bli brukt av ESM for å finne brukernes prioritet. Andre deler av Elvira kan senere ønske å bruke denne databasen til f.eks. autentifisering av brukere.

Brukerdatabasen er implementert veldig enkelt, og modulene som bruker den må ha tilgang til den samme disken som passordprogrammet befinner seg på. Dersom dette ikke er tilfelle, er det ikke noe problem å sette opp f.eks. UNIX-programmet **rdist(1)** til å distribuere filene med jevne mellomrom.

I denne prøveimplementasjonen er endel av informasjonen uviktig, som f.eks. avdeling og organisasjon, og denne vil bli utelatt. (Det vil være enkelt å legge til slik informasjon senere).

[1] Ikke alle formatene ISO 8601 spesifiserer støttes.

7.8 Eksempler på bruk

Her vises diverse eksempler på kjøring av testprogrammene nevnt ovenfor. Disse resultatene er tatt med for å vise at viktige deler av ESM fungerer, og fungerer i tillegg som en illustrasjon til det som har blitt sagt i dette og tidligere kapitler.

7.8.1 Import av data i ESM

Her er et eksempel hvor det blir registrert et bånd med 100000 blokker og blokkstørrelse 32768 byte (da blir kapasitet rundt 3.2GB) ved hjelp av **ttapadmin**. Deretter lagres fire arkiver med henholdsvis 1, 2, 2 og 10 båndfiler, der hver båndfil er 1MB[2]. Dette gjøres ved hjelp av **timport**.

```
% ttapeadmin.exe 100000 32768 y
% timport.exe 1 1 y /tmp/img.tmp
% timport.exe 2 2 y /tmp/img.tmp
% timport.exe 3 2 y /tmp/img.tmp
% timport.exe 4 10 y /tmp/img.tmp
```

7.8.2 Metadata

I dette eksempelet kalles **tmetadb** for hver av de tre typene metakataloger. Vi antar at ESM var tom da vi utførte eksempelet i avsnitt 7.8.1. Først ser vi på innholdet i databasen for informasjon om hvert bånd:

```
% tmetadb.exe t
TID      LBLW     CAP      BLFREE   BLSIZ

       1      465   100000    99535    32768
```

Forkortelsene som er brukt:

- TID = bånd-identifikator
- LBLW = adressen til siste blokk som er skrevet på båndet
- CAP = kapasiteten til båndet
- BLFREE = antall blokker ledige
- BLSIZ = størrelsen på hver blokk

Databasen med metainformasjon om arkivene inneholder informasjon om arkivene og filene inne i arkivene. Først ser vi på en listing av arkivene som finnes i ESM:

```
% tmetadb.exe a
AID   ADDR     FILES TID   SIZE     BLOCKS     DATE

    1        1     1     1  1000000        31   1997-03-12 17:52:22Z
    2       32     2     1  2000000        62   1997-03-12 18:01:42Z
    3       94     2     1  2000000        62   1997-03-12 18:37:24Z
    4      156    10     1 10000000       310   1997-03-12 18:52:23Z
```

Forkortelsene som er brukt:

- AID = arkiv-identifikator

[2] 1 000 000 bytes, ikke 1 048 576 bytes

- ADDR = addressen arkivet har på bånd
- FILES = antall filer i arkivet
- TID = bånd-identifikator
- SIZE = størrelse på arkiv i bytes
- BLOCKS = antall blokker arkivet bruker på bånd
- DATE = tiden arkivet ble lagret (i UTC-tid), formattert etter ISO 8601-standarden[3]

Deretter lister vi alle båndfilene:

```
% tmetadb.exe f
AID  TFNO  ADDR    SIZE

  1    0        2  1000000
  2    0       33  1000000
  2    1       64  1000000
  3    0       95  1000000
  3    1      126  1000000
  4    0      157  1000000
  4    1      188  1000000
  4    2      219  1000000
  4    3      250  1000000
  4    4      281  1000000
  4    5      312  1000000
  4    6      343  1000000
  4    7      374  1000000
  4    8      405  1000000
  4    9      436  1000000
```

Forkortelsene som er brukt er:

- AID = arkiv-identifikator
- TFNO = båndfil-nummer (nummerert fra 0 til antall båndfiler -1)
- ADDR = absolutt blokkaddresse til båndfil (ikke relativ til starten av arkivet!)
- SIZE = størrelsen (i bytes) av båndfil.

[3] UTC-tid er det samme som GMT-tid. Bokstaven Z står for Zulu, som de militære bruker om UTC-tid.)

Kapittel 8

Målinger

Dette kapittelet presenterer de målingene som er utført ved bruk av ESM. Dataoverføring med tertiær videopumpe og Elvira Tape Server (ETS) har blitt undersøkt. Målingene ble i sin helhet utført med Tandberg Data MLR-1 båndstasjon på en Axil 320 HyperSPARC-maskin som kjørte SunOS 5.4 (Solaris 2.4). En annen maskin av samme type ble brukt for å kjøre videopumper og denne var også disktjener. Alle dataoverføringer over nettverk gikk via et 155 Mbit/s ATM-nettverk. Resultatgrafene tar stor plass, og er derfor listet etter hverandre i slutten av dette avsnittet.

Begrensninger

På grunn av manglende rensebånd kunne ikke båndstasjonen renses før målingene tok til, noe som førte til at den ikke ytte sitt beste under målingene. Overføringsraten for skriving til bånd var bare to tredjedeler av hva den var målt til da stasjonen var «ren». Leseraten var enda dårligere, omtrent en tredjedel av maksimal overføringsrate. Det verste var at overføringsratene antok mer eller mindre tilfeldige verdier fra forsøk til forsøk.

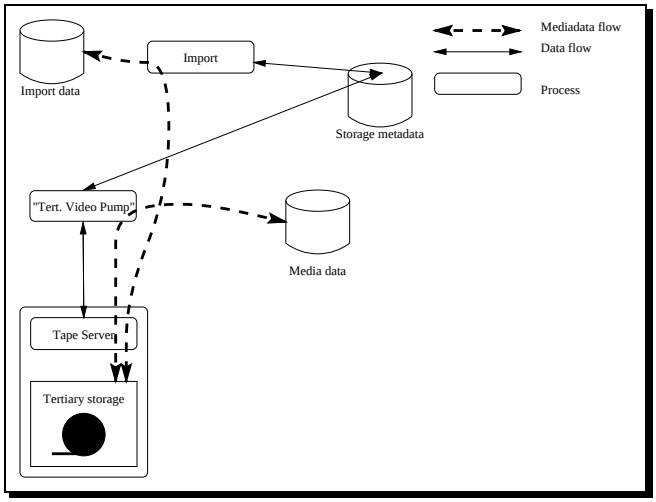
Enheter

I dette kapittelet brukes det *ikke* metriske verdier for forstavelsene kilo-, mega- og giga foran byte. Istedet brukes $1kB = 2^{10}B$, $1MB = 2^{20}B$, osv. Dette er gjort fordi ESM opererer med slike verdier for størrelse på filer.

8.1 Aksesstid for arkiver på forskjellige blokkadresser

Som nevnt flere ganger i kapittel 4 er aksesstider svært viktige for ytelsen til en videoarkivtjener som Elvira/ESM. I dette avsnittet beskrives en måling der man hadde et oppsett lik det som er vist i figur 8.1.

Førti arkiver ble lagret etter hverandre ved hjelp av **timport.exe**, beskrevet i avsnitt 7.2. Hvert arkiv bestod av en båndfil på 10MB. Vanligvis vil ikke mediafiler være så små. 10MB video med en båndbredde på 2Mbit/s vil bare vare i 40 sekunder. Med en blokkstørrelse på 32kB, vil et arkiv bruke 320 blokker, pluss en blokk for arkivheader. I dette eksperimentet er det imidlertid aksesstid som er i fokus, og for å få et rimelig antall målinger langs hele båndlengden ble denne størrelsen valgt. Et bånd til MLR-1 har en kapasitet på 13GB, og det er hele 144 spor på et bånd. To og to spor blir skrevet av gangen, så en båndlengde har en kapasitet på omtrent 180MB, eller 5500 32kB blokker. 40 arkiver vil utgjøre 12840 blokker, og vil derfor dekke to båndlengder, eller en hel «trekant» og litt til, slik som vist i figur 8.2. Mellom alle målinger ble båndet spolt tilbake, slik at



Figur 8.1: Konfigurasjon ved måling av aksesstid

det er aksesstider fra begynnelsen av båndet som er målt.

8.1.1 Resultat

I figur 8.2 ser man resultatet av målingene. Som man kan se av figuren, har grafen en trekantform. Dette skyldes selvsagt at dataene lagres på et serpentinbånd, og mønsteret ligner det som ble funnet for TDC 4220, slik som vist i avsnitt 4.1, side 30.

Et interessant fenomen er at aksesstidene øker så å si helt lineært når kurven stiger, mens kurven er mer uryddig når den minsker. Ut fra figuren er det vanskelig å se hva dette skyldes, men dette blir diskutert videre i neste delavsnitt.

Selv om målingene av overføringsrate ikke er hovedtema for dette avsnittet, kan man gjøre en interessant observasjon ved skifte av spor. Ved sporskifter må nemlig båndstasjonen reposisjonenere lesehodet, slik at det kan lese på neste spor. Dette tar litt tid, og dermed øker overføringstiden for blokker som går over et sporskifte.

8.1.2 Sammenligning med andre målinger

Figur 8.3 viser aksesstidene for ESM plottet mot aksesstider målt direkte mot båndstasjon, uten bruk av ETS (Andersen 1997). Som det går frem av figuren, er faktisk søkingen et par sekunder raskere enn ved direkte aksess mot stasjon. Dette er antakeligvis fordi tilbakespoling ved hjelp av ETS gjør at båndet blir klargjort for lesing eller spoling. Ved eksperimentene direkte mot bånd, ble det bare gjort en ren tilbakespoling, uten klargjøring for lesing/spoling.

Det merkelige mønsteret observert forrige delavsnitt er nå lettere å forstå. De detaljerte målingene gjort av Andersen (1997) viser nemlig at det kurven for minskende aksesstider slettes ikke er så jevn som den for økende aksesstider. Kurven følger et sagtannmønster, og dette påvirket naturlig nok også målingene på tertiær videopumpe.

8.1.3 Standardavvik og konfidensintervaller

Aksesstider mot de 40 arkivene ble målt 8 ganger, ved å repetere prosessen beskrevet i innledningen av dette avsnittet. Dette var en svært tidkrevende prosess, og oppgaven ble utført automatisk av

et perl-script om natten. Et plott av enkeltmålingene finnes i figur 8.4. Det går fram av figuren at variasjonen fra måling til måling var svært liten, siden punktene ligger svært nære hverandre. Resultatet av målingen ble også brukt for å finne standardavvik og konfidensintervaller for aksesstider. Standardavviket til en måleverdi, sier noe om hvor mye den varierte fra måling til måling, kan finnes ved å bruke følgende formel:

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x - \overline{x})^2}{n-1}}$$

s er standardavviket x er måleverdien, n er antall målinger og $\overline{x}$ er gjennomsnittsverdi for x, dvs:

$$\overline{x} = \frac{\sum_{i=1}^{n} x}{n}$$

Konfidensintervaller angir en øvre og en nedre verdi som det er forventet at en viss prosentandel av målingene vil havne innenfor. Variansen ser ut til å være rimelig tilfeldig, og man kan derfor gå ut fra at aksesstidene er normalfordelte. Da har man følgende uttrykk for et $(1-\alpha)100\%$ konfidensintervall (Dougherty 1990):

$$\left(\overline{x} - t_{\frac{\alpha}{2},n-1} \cdot \frac{s}{\sqrt{n}}, \overline{x} + t_{\frac{\alpha}{2},n-1} \cdot \frac{s}{\sqrt{n}}\right)$$

Variablene n og s og $\overline{x}$ er de samme som over. α er et tall mellom 0 og 1 som sier hvor sikkert intervallet skal være. $\alpha = 0.05$ gir derfor et 95% konfidensintervall. $t_{\frac{\alpha}{2},n-1}$ er en tabulert verdi for den såkalte *Student t-distribusjonen*. For våre målinger vil vi ha $t_{0.025,7} = 2.365$ for et konfidensintervall på 95%.

Det ble laget et perl-script kalt **statistics.pl** for å beregne disse verdiene. Utskriften under består av kolonner for blokknummer; antall målinger; gjennomsnitts-, minimal- og maksimalverdier; standardavvik; og øvre og nedre grenser for 95% konfidensintervall. Ikke alle målingene er vist. Sekunder er brukt som måleenhet, bortsett fra for standardavvik, som er målt i millisekunder.

```
% statistics.pl variance.log
#blk    n mean     min      max      stdev  lb95     hb95
1       8  0.0     0.0      0.0      4.8    0.0      0.0
322     8  6.9     6.6      7.2      159.5  6.7      7.0
643     8  13.7    13.4     14.2     226.9  13.5     13.9
964     8  20.9    20.4     21.8     428.8  20.6     21.3
1285    8  27.3    27.0     27.4     150.2  27.1     27.4
1606    8  34.2    33.7     35.1     348.2  33.9     34.5
...
4495    8  95.6    95.2     95.8     154.4  95.5     95.7
4816    8  102.3   102.0    102.6    183.4  102.2    102.5
5137    8  109.4   109.1    109.8    188.6  109.2    109.5
5458    8  119.0   118.6    119.9    366.3  118.7    119.4
5779    8  112.9   112.3    113.6    389.5  112.5    113.2
6100    8  110.8   110.5    111.1    202.2  110.6    110.9
6421    8  100.3   99.8     101.0    305.6  100.1    100.6
6742    8  98.6    98.1     99.3     380.9  98.3     98.9
7063    8  88.1    87.4     88.7     399.7  87.7     88.4
...
9952    8  27.3    26.9     27.8     298.5  27.0     27.5
10273   8  16.6    16.2     16.9     220.8  16.4     16.8
10594   8  14.7    14.4     15.2     200.5  14.5     14.9
10915   8  4.2     3.7      4.8      339.2  3.9      4.5
...
```

I figur 8.5 er standardavviket plottet mot blokkadresser. Det er vanskelig å trekke noen konklusjon ut fra målingene, annet enn at den ser rimelig tilfeldig ut.

8.2 Overføringsrate for store arkiver

Forrige avsnitt konsentrerte seg om aksesstider. Dette vil ta for seg overføringsrater ved forskjellige konfigurasjoner av ESM. Oppsettet for målingene er nøyaktig det samme som i forrige avsnitt. Som nevnt i innledningen av dette kapittelet, er det tvil om verdien av disse målingene, på grunn av redusert kapasitet for båndstasjonen. Under målingene gjort her, varierte overføringsraten for kontinuerlig skriving mellom 0.80MB/s og 1.3MB/s fra den ene dagen til den andre.

I dette forsøket studeres overføringsrater for et 300MB stort arkiv. Målinger på overføringsrate er gjort på forskjellige arkiv med 1, 2, 4, 6, 10, 15 og 30 båndfiler. Resultatene er vist i tabell 8.1 og i figur 8.6. Som det går fram av resultatene, er det stor forskjell mellom skrivetid og lesetid. Dette skyldes nok i stor grad at båndstasjonen måtte bruke lang tid på feilkorrigering på grunn av urenheter på lesehodet.

Forventet form på kurven hadde vært at lesetidene ble høyere jo flere båndfiler det var i hvert arkiv. Den lille pausen mellom hver lesing av et båndarkiv gjør at båndstasjonen må stoppe opp. Dette er kostbart, fordi den må reposisjonere seg før den kan fortsette (Silberschatz og Hillyer 1996*c*). Det virker som om dette problemet ikke påvirker resultatene i nevneverdig grad. Det er i det hele tatt vanskelig å lese noe fornuftig ut av kurven.

Antall båndfiler	Størrelse på båndfil	Skrivetid	Lesetid
1	300MB	462s	236s
2	150MB	452s	231s
4	75MB	451s	249s
6	50MB	453s	286s
10	30MB	440s	286s
15	20MB	468s	266s
30	10MB	463s	247s

Tabell 8.1: Måleresultater for 300MB arkiv med varierende antall filer

Disse resultatene er beklageligvis ikke særlig entydige. Det burde ha vært en klar tendens til at overføringstidene for lesing burde ha økt når arkivet bestod av økende antall filer.

8.2.1 Sammenligning med andre målinger

For å kunne sammenligne ESMs verdier for overføringsrate mot overføringsrater målt direkte mot båndstasjon, uten bruk av ETS, er det blitt utført målinger av undertegnede ved hjelp av et test-program utviklet av Rune Sætre. Det ble utført 5 målinger lesing av en 300MB stor fil. Resultatene var: 434, 435, 436 og 434 sekunder. Disse målingene ble utført rett etterhverandre, noe som kan forklare den lille variasjonen. Målinger utført uten loggføring dagen før gav resultater på rundt 500 sekunder, så det er tydelig at båndstasjonen har problemer på grunn av den manglende rensingen.

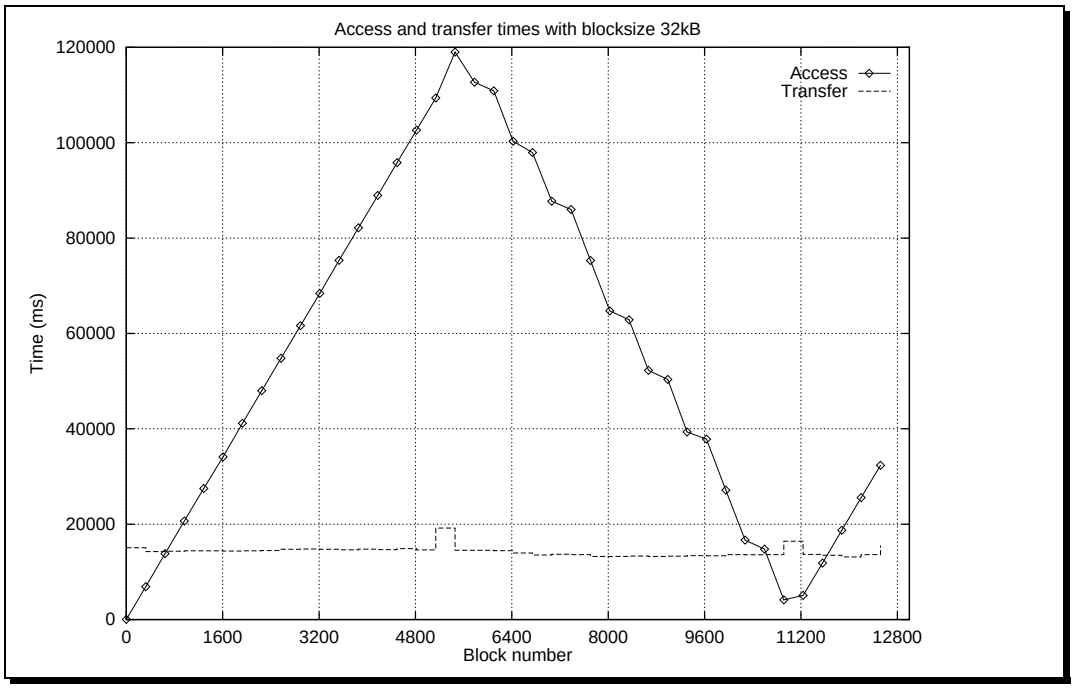
8.3 Oppsummering

ETS og ESM utnytter aksess-hurtigheten til båndstasjonen like effektivt som om båndstasjonen skulle blitt brukt direkte. Målekurvene for aksesstider for ESM og målekurvene for aksesser direkte mot båndstasjon følger hverandre tett, noe som viser at når systemet er lite belastet, vil ikke systemet påføre noen ytterligere forsinkelser på aksesstid i forhold til båndstasjon. Variasjonene i aksesstid fra måling til måling var svært liten. Dette betyr at det er mulig å estimere aksesstider med stor grad av nøyaktighet for en MLR-1 båndstasjon.

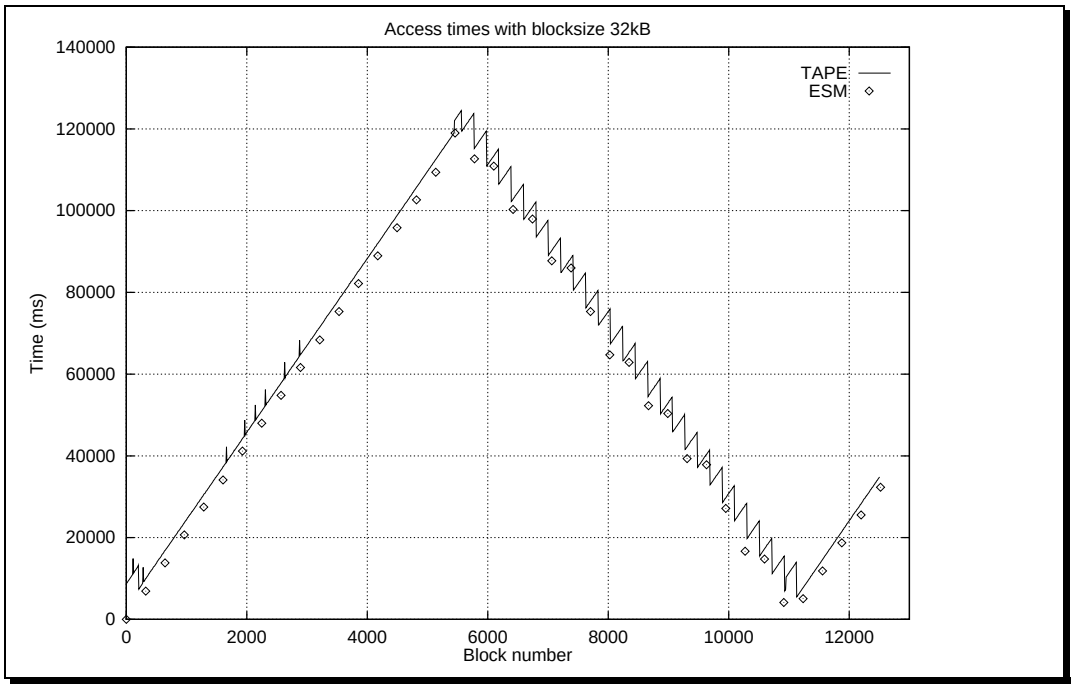
Målingene på overføringsrate ga svært uklare resultater, og problemet med båndstasjonen må nok

ta skylden for dette. Selv om målingene ikke gav noe klart svar, så er bare det at de ble gjennomført uten store problemer et bevis på at ETS, tertiær videopumpe og importfunksjon ser ut til å være rimelig stabile, selv om de ennå er på test-stadiet.

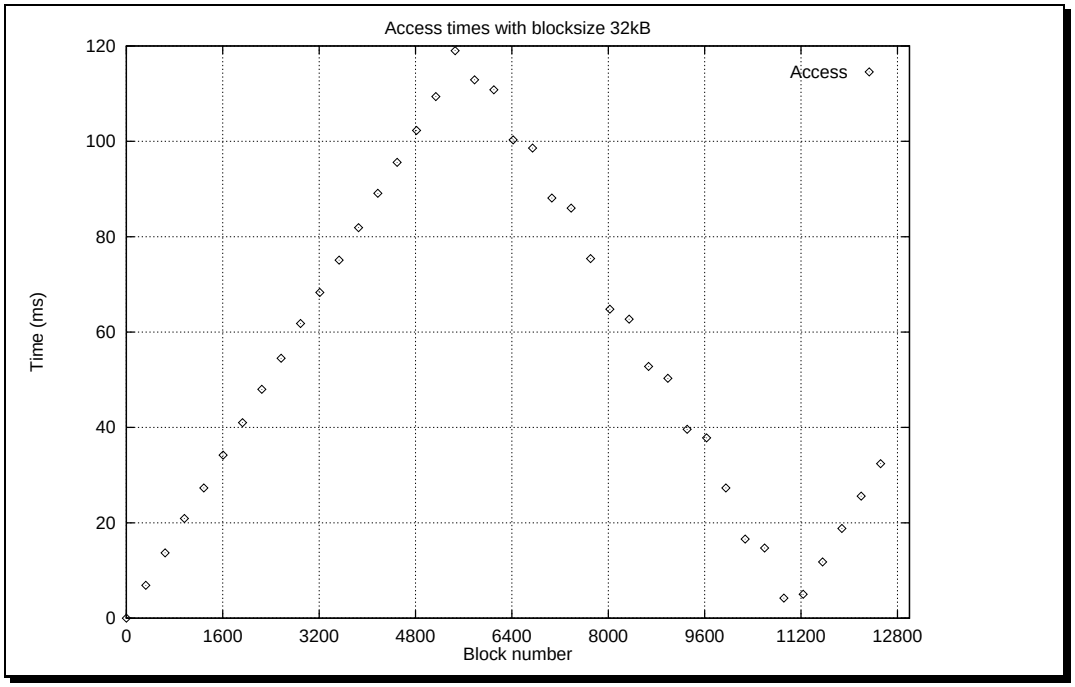
8.4 Grafer



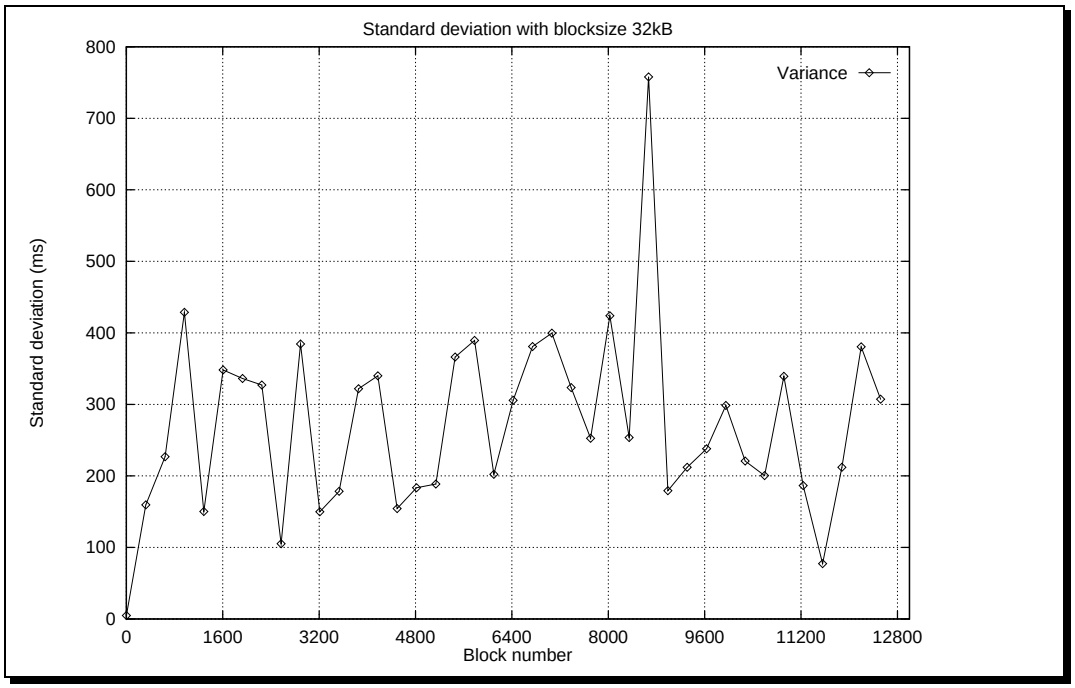
Figur 8.2: Aksesstider mot 40 10MB arkiver med en båndfil



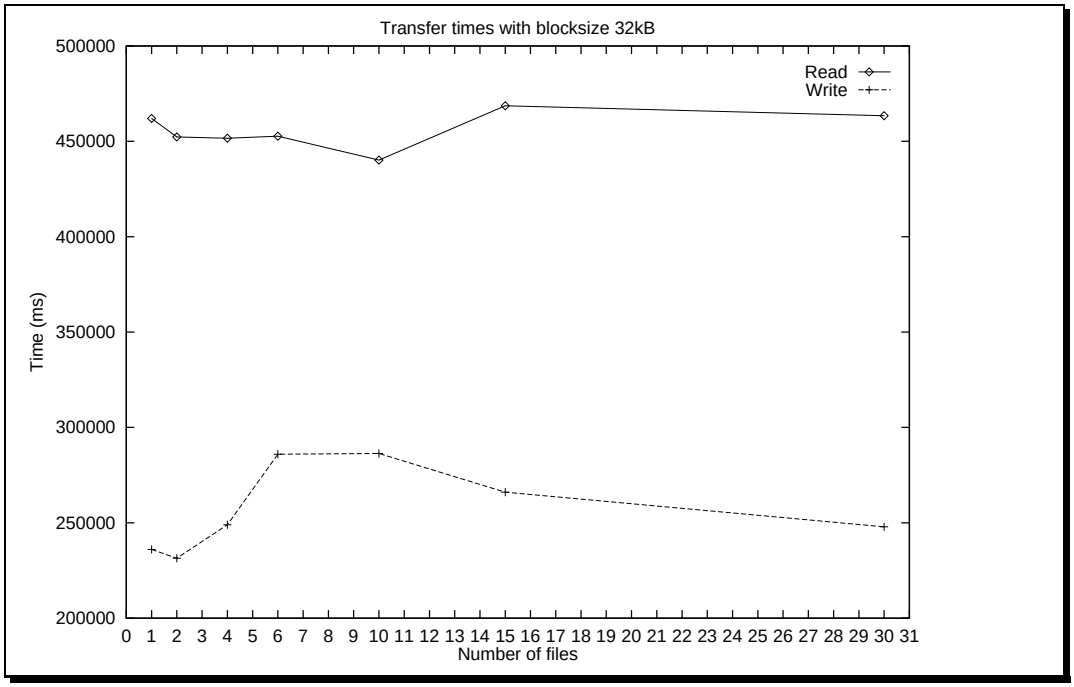
Figur 8.3: Målinger med ESM vs. direkte bruk av båndstasjon



Figur 8.4: Aksesstider målt 8 ganger



Figur 8.5: Standardavvik for aksess mot spesifikk blokk



Figur 8.6: Aksesstider mot ett 300MB arkiv med varierende antall filer

Kapittel 9

Konklusjon

Hva er oppnådd gjennom arbeidet med denne hovedoppgaven? Dette kapittelet forsøker å gi et svar på dette. Først oppsummeres utført arbeid, deretter følger en konklusjon basert på erfaringene fra dette arbeidet. Videre følger et avsnitt om spesielle erfaringer som undertegnede mener leseren av denne rapporten kan dra nytte av. Til slutt kommer noen forslag til videre arbeid relatert til tjenere for digitale videoarkiv.

9.1 Arbeid utført

Det er utført et studium av masselagersystemer, videotjenere og videoarkiv, standarder for digital video og audio og forskningsresultater for tertiære lagringsmedier. Det har funnet sted en idéutveksling med Norsk Rikskringkasting angående bruk av digitale videoarkivtjenere. Denne informasjonen er brukt som grunnlag og inspirasjon for det videre arbeidet. Dette er utført i løpet av tidsperioden avsatt til denne oppgaven:

- En modell for en digital videoarkivtjener er utviklet. Modellen er en gjennomgang av forskjellige parametere for et system som først består av en enkelt båndstasjon med ett bånd og siden blir utvidet med flere bånd og til slutt flere båndstasjoner. Det gis en beskrivelse av de enkelte parametrenes betydning for ytelsen til tjeneren, og avhengigheter blir diskutert. Til slutt blir tjenerens omgivelser diskutert.
- Det er laget en konstruksjon for en modul som skal gjøre Elvira2 i stand til å benytte digitale databånd for lagring av mediadata. Viktige deler av denne konstruksjonen er implementert.
- Ytelsesmålinger av visse deler av implementasjonen har blitt gjennomført. Det ble fokusert på aksess- og overføringstider for lasting av materiale til harddisk, dette er to parametere som ble spesielt fremhevet som viktige for ytelsen til et digitalt videoarkiv med lagring på bånd. Grunnen til dette er at disse parametrene er viktige for totalytelsen til systemet og responstid for brukerne. Måleresultatene ble analysert og diskutert.

9.2 Konklusjon

Eksisterende store arkivsystemer fokuserer på kostnadseffektivitet for lagring, mens rene videotjenere er fokusert på lavest mulig kostnad for båndbredde. En videoarkivtjener er et forsøk på å utnytte det beste fra begge teknologier.

Modellen for en videoarkivtjener bidrar med teoretisk bakgrunn for de valgene som er truffet i senere deler av oppgaven. Den kan brukes for å få bedre innsikt i hvilke faktorer som er av avgjørende betydning for ytelse i et slikt system. En av de viktigste konklusjonene fra denne delen av rapporten

er at overføringsrate og aksesstider mot båndbiblioteket der videofilmene er lagret er svært viktige for systemytelsen.

Systemet som har blitt utviklet, Elvira Storage Manager, er ment å gjøre videotjeneren Elvira i stand til å fungere som en videoarkivtjener. ESM vil bestå av fire hovedmoduler som sammen er ansvarlige for å flytte video mellom bånd og harddisk etter brukernes behov.

Brukernes forespørsler om henting av videofilm vil organiseres av en ressursfordeler som lager kjøreplaner for overføring av video. Det er tatt i bruk en egenutviklet algoritme for ressursfordeling som benytter seg av det faktum at brukere av en videoarkivtjener vil ha forskjellige behov i forbindelse med innhenting av materiale. Ressursfordeleren støtter blant annet bestilling av video lang tid i forveien og direkte henting av video. Det er innført mekanismer for å fordele ressurser mellom ulike forespørsler på en fornuftig og rettferdig måte.

Begrenset lagringskapasitet på harddisker nødvendiggjør streng kontroll med forbruk av diskplass. Harddiskene vil bli brukt som et buffer for videomaterialet som primært ligger på bånd. Når nytt materiale hentes inn fra bånd, må eksisterende materiale på harddiskene slettes for å gjøre plass. Bufferhåndtereren vil benytte seg av en modifisert LRU-algoritme som kaster ut filmer som ikke har blitt benyttet på lang tid. Modifiseringen består i at brukere som har behov for det skal få muligheten til å låse en eller flere viktige videofilmer i harddiskbufferet, slik at de ikke vil skiftes ut.

Selve overføringen av data gjøres av en modul kalt tertiær videopumpe. Denne modulen benytter et grensesnitt mot en båndstasjonstjener for å lese data fra et delt minnebuffer og sender dataene videre til Elviras videopumper.

For å holde orden på det som er lagret i båndbiblioteket er det utviklet en modul for lagring av metadata på harddisk. Metadatabasene vil forsyne resten av modulene i ESM med den informasjonen de trenger for å fungere. Metadataene blir blant annet brukt til å registrere hvilke videofilmer som befinner seg på hvilke bånd. Det blir blant annet registrert størrelse, båndtilhørighet og startadresse på bånd for hver videofilm. Et eksempel på bruk av metadata er den tertiære videopumpen, som kun får en identifikator for videofilmen som skal hentes. Den tertiære videopumpen slår opp i metadatabasen for arkivinformasjon for å finne ut hvor den kan finne igjen filmen.

Det er utført en konstruksjon av de fire hovedmodulene, og den tertiære videopumpen og metadatabasene er ferdig implementert. Implementasjon av de to andre modulene er påbegynt og er et godt stykke på vei. Det har også blitt implementert endel nyttige støttemoduler for å lette arbeidet med ESM-modulene. Det står fortsatt igjen en del arbeid med systemintegrasjon, kontroll og kommunikasjon med andre moduler i Elvira.

Målinger av aksesstid og overføringsrater for systemet er målt ved å instrumentere den tertiære videopumpen og deretter kjøre forskjellige testprogrammer. Det ble vist at aksesstiden er svært avhengig av videofilmens plassering på båndet. Aksesstider følger et trekant-mønster for stigende blokkadresser, der bunnen ligger på under ett sekund og toppene på rundt 120 sekunder, når hver måling ble gjort fra starten av båndet. Målingene viste også at aksesstider for en og samme blokkadresse varierer svært lite fra gang til gang. Dette kan utnyttes ved estimering av aksesstider i et mer fullstendig system for digitale videoarkiv. Målingene av overføringsrate ble dessverre forstyrret av tekniske problemer og ga ikke entydige resultater.

9.3 Videre arbeid

Dette avsnittet tar for seg mulige videreføringer av arbeidet gjort i denne hovedoppgaven. Forslagene er fokusert på tjenere for digitale videoarkiv og kan tenkes brukt som inspirasjon for senere prosjekter og hovedoppgaver på IDI.

9.3.1 Ferdigstilling av ESM og systemevaluering

Dagens ESM er ikke ferdigstilt. De viktigste byggeklossene er på plass, men selve systemet er ikke i stand til å brukes i Elvira slik det er i dag. Kontroll-, protokoll- og tjener-modul må implementeres og Elvira med ESM bør testes på systemnivå og målinger av systemet med forskjellige belastningspådrag bør utføres.

Det må også gjøres endringer av andre moduler i Elvira før ESM kan fungere. Det viktigste arbeidet som står igjen her er å endre Command Server slik at den kan kommunisere med ESM og legge til funksjonalitet i videopumpene slik at de kan lagre data, isteden for å kun lese, slik som i dag.

9.3.2 Lagringsformater for digital video og audio

Moduler for å gjøre ESM i stand til å tolke forskjellige lagringsformater vil være av interesse. Slike må eksistere dersom ESM skal være i stand til å utnytte indeksfiler ved aksesser mot video lagret på bånd. ESM bør ikke gjøres avhengig av et spesielt lagringsformat, så det bør utvikles et generisk grensesnitt for behandling av videoinformasjon, der man kan sette inn innstikk-moduler for de forskjellige formatene.

Innstikk-modulene må være i stand til å (1) generere indekser for et format og (2) slå opp i disse indeksene for å konvertere mellom tidskoder og adresser (i bytes eller blokker) i mediadata.

9.3.3 Kapasitets- og ytelsesvurdering

Modell-kapittelet i denne rapporten identifiserer en rekke parametere og diskuterer deres relevans for ytelsen til en videoarkivtjener. Det kan være interessant å foreta en grundig matematisk og statistisk analyse på ytelse, pålitelighet og kostnader for et en slik tjener. En slik analyse bør munne ut i konkrete tall og regler som gjør organisasjoner i stand til å sette sammen en tjener på papiret uten å måtte prøve seg fram ved å prøve og feile og dermed kaste bort store mengder kapital.

9.3.4 Distribuerte videoarkivtjenere

Det kan være aktuelt å la Elvira/ESM får en distribuert natur, slik som Berkeley VODS i avsnitt 3.2.5, side 21. Dette er inspirert av NRKs situasjon, der man har eksisterende manuelle arkiver i hvert distriktskontor og sentrale arkiver ved hovedkvarteret i Oslo. En utvikling av en modell for en slik distribuert løsning, samt konstruksjon og implementasjon og testmålinger burde gi interessante resultater.

9.4 Andre erfaringer

Her følger en beskrivelse av endel positive erfaringer undertegnede har gjort under arbeidet med hovedoppgaven:

- Under arbeidet med forstudiet viste søkeverktøyet INSPEC på universitetsbiblioteket seg å være svært nyttig. Brukt sammen med søkemaskinen AltaVista på Verdensveven førte dette til at det var mulig å laste ned relevante vitenskapelige artikler på et øyeblikk. Verktøyet NCSTRL (Networked Computer Science Technical Reports Library) som befinner seg på `www.ncstrl.org` var også til stor hjelp. Det er undertegnedes mening at IDI burde delta i dette samarbeidsprosjektet.
- Programmering i C++ må skje med omhu. Bøkene «Effective C++» (Meyers 1992), «More Effective C++» (Meyers 1996) tar for seg hva man bør og hva man ikke bør gjøre i C++.

Disse bøkene kan anbefales på det varmeste. «Designing and Coding Reusable C++» (Carroll og Ellis 1995) gir gode råd om bruk av parametriserte klasser og bibliotekskriving. Verktøyet Purify (Pure Atria 1997) var til stor hjelp for oppsporing av blant annet minnelekkasjer. Store mengder nyttig informasjon er funnet på Verdensveven og på USENET.

- Programmeringen av støttemodulene (GDBM og isotime) ble gjort tidlig. Dette resulterte i at disse klassene var gjenstand for høy grad av gjenbruk under implementasjonen av ESM-spesifikke klasser. Koden ville utvilsomt blitt mye større og komplisert uten bruk av disse klassene.

Bibliografi

Andersen, T. M. (1997), 'Målinger av aksesstid på MLR-1'. Unpublished graphs.

Appelbaum, R., Cline, M. og Girou, M. (1997), 'The CORBA FAQ', WWW.
*http://www.cerfnet.com/ mpcline/Corba-FAQ/

Asami, S., Talagala, N., Anderson, T., Lutz, K. og Patterson, D. (1995), The Design of Large-Scale, Do-It-Yourself RAIDs. Draft.
*http://now.cs.berkeley.edu/Td/td.html

Bell, A. E. (1996), 'Next-Generation Compact Discs', *Scientific American* **285**(1), 28–32. The author was the chairman of the DVD Technical Working Group.

Bolosky, W. J., Joseph S. Barrera, I., Draves, R. P., Fitzgerald, R. P., Gibson, G. A., Jones, B. B., Levi, S. P., Myhrvold, N. P. og Rashid, R. F. (1996), The Tiger Video Fileserver, *in* 'Proceedings of the Sixth International Workshop on Network and Operating System Support for Digital Audio and Video', Microsoft Research, Microsoft Corporation, IEEE Computer Society, pp. 97–104. Zushi, Japan.
*ftp://ftp.research.microsoft.com/pub/tech-reports/Winter95-96/TR-96-09.ps

Brassard, G. og Bratley, P. (1987), *Algorithmics - Theory and Practice*, Prentice Hall, chapter 3, pp. 92–100.

Bratbergsengen, K. (1996), *Filsystemer - Lagring og behandling av store datamengder*, TAPIR.

Carroll, M. D. og Ellis, M. A. (1995), *Designing and Coding Reusable C++*, Addison-Wesley.
*http://heg-school.aw.com/cseng/authors/carroll/designC++/designC++.html

Caulk, P. M. (1995), The Design of a Petabyte Archive and Distribution System for the NASA ECS Project, Technical report, NASA.
*http://eospso.gsfc.nasa.gov/eos_homepage/eosdis.html

Chervenak, A. L., Patterson, D. A. og Katz, R. H. (1995), Choosing the Best Storage System for Video Service, *in* 'Multimedia '95, San Francisco, California, USA', ACM, pp. 109–119.

Cormen, T. H., Leierson, C. E. og Rivest, R. L. (1990), *Introduction to Algorithms*, MIT Electrical Engineering and Computer Science Series, MIT Press, chapter 17, pp. 345–353.

Dougherty, E. E. (1990), *Probability and Statistics for the Engineering, Computing and Physical Sciences*, 1 edn, Prentice Hall, chapter 6, pp. 283–295.

Drapeau, A. L. og Katz, R. H. (1993), Striped tape arrays, *in* 'Twelfth IEEE Symposium on Mass Storage Systems', University Of California, Berkeley, IEEE Computer Society Press, pp. 257–265.

Dybvik, G. A. (1996), Editering av MPEG video, Technical report, IDT/NTNU.

Elmasri, R. og Navathe, S. (1994), *Fundamentals of Database Systems*, Addison Wesley. ISBN 0-8053-1753-8.

Exabyte Corporation (1997), 'Product information', WWW.
*http://www.exabyte.com/home/products.html

Federighi, C. og Rowe, L. A. (1994), A Distributed Hierarchical Storage Manager for a Video-on-Demand System, *in* 'Storage and Retrieval For Image and Video Databases II, San Jose, California, USA', The International Society for Optical Engineering.
*http://bmrc.berkeley.edu/projects/vods/

Ford, D. A., Morris, R. J. T. og Bell, A. E. (1996), Redundant Arrays of Independent Libraries (RAIL): A Tertiary Storage System, *in* 'Compcon '96 – Forty-First IEEE Computer Society International Conference', IBM Almaden Research Center, IEEE Computer Society Press, pp. 280–285.

Fournier, M. G. (1997), 'Postgresql', WWW.
*http://www.postgresql.org/

Hetland, M. (1996), Videotools for the world wide web, Master's thesis, IDT/NTNU.
*http://www.idt.unit.no/IDT/grupper/DB-grp/report_diplomas/diplomas.html

Hjelsvold, R. (1995), VideoSTAR - A Database For Video Information Sharing, PhD thesis, NTH/IDT.

Horowitz, E. og Sahni, S. (1978), *Fundamentals of Computer Algorithms*, Computer Science Press – Inc. Computer Software Engineering Series, Pitman Publishing Inc., chapter 4, pp. 161–168. ISBN 0-13-023243-2.

Iomega Corporation (1996), 'Jaz Technical Specs'.
*http://www.iomega.com/product/jaz/jazspec.html

Kernighan, B. W. og Ritchie, D. M. (1988), *The C programming language*, Prentice Hall software series, 2 edn, Prentice Hall. ISBN 0-13-110362-8.

Langørgen, S. (1994), Elvira - Eksperimentell Videotjener for ATM, Master's thesis, NTH/IDT.
*http://www.idt.ntnu.no/IDT/grupper/DB-grp/report_diplomas/diplomas.html

Laursen, A., Olkin, J. og Porter, M. (1994), Oracle Media Server: Providing Consumer Based Interactive Access to Multimedia Data, *in* 'ACM SIGMOD, Minneapolis, Minnesota', Oracle Media Server Development.

Lo, V. (1997), 'A Beginners Guide for MPEG-2 Standard', WWW.
*http://www.ee.cityu.edu.hk/~edap064/mpeg/BeginnersGuideForMPEG2Standard.html

Meyers, S. (1992), *Effective C++*, Professional Computing Series, Addison-Wesley.
*http://www.awl.com/cp/meyers-effective.html

Meyers, S. (1996), *More Effective C++*, Professional Computing Series, Addison-Wesley. ISBN 0-201-63371-X.
*http://www.awl.com/mec++.html

Nelson, P. A. og Gaumond, P. (1995), *GNU Database Manager*.

N.N. (1997), 'JPEG image compression FAQ', USENET.
*http://www.cis.ohio-state.edu/hypertext/faq/usenet/jpeg-faq/top.html

Noreld, M. (1996), 'Fjernsynsarkivet pr. 01.01.96'. En statusrapport for hva som er lagret i NRKs arkiv.

Oracle Corporation (1996), 'Oracle Media Server - Data Sheet', WWW.
*http://www.oracle.com/products/media_server/datasheets/datasheet.html

Pure Atria (1997), 'Purify product overview', WWW. Pure Atria is a company selling software quality tools.
*http://www.pureatria.com/

Quantum Corporation (1997), 'The DLT 7000 Tape Drive: A Quantum White Paper', WWW.
*http://www.quantum.com/products/whitepapers/dlt/DLT7000.HTM

Robbins, K. A. og Robbins, S. (1996), *Practical UNIX Programming*, Prentice-Hall.

Rowe, L. A., Boreczky, J. S. og Eads, C. A. (1994), Indexes for User Access to Large Video Databases, *in* 'Storage and Retrieval For Image and Video Databases II, San Jose, California, USA'.
*http://bmrc.berkeley.edu/projects/vods/

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. og Lorensen, W. (1991), *Object-Oriented Modeling and Design*, Prentice Hall.

Saha, K. K. (1996), Lagring av store mengder data. Internal report at IDT.

Sample, M. og Joop, R. (1995), *Snacc 1.2rj: A High Performance ASN.1 to C/C++/IDL Compiler*, 1.2 edn, University of British Columbia and Forschungszentrum Informationstechnik GmbH, Germany. ftp://ftp.fokus.gmd.de/pub/freeware/snacc/.

Sandstå, O. (1997), 'Miscellaneous notes on design of Elvira2'.

Sandstå, O., Midtstraum, R. og Langørgen, S. (1996), Design and Implementation of the Elvira Video Server, *in* 'NIK 94', IDT/NTNU.

Savatier, T. og Fogg, C. (1997), 'MPEG Starting Points', WWW. The authors are members of the MPEG.
*http://www.mpeg.org/

Seagate (1997), 'Product information', WWW.
*http://www.seagate.com/disc/disctop.shtml

Sethi, R. (1989), *Programming Languages – Concepts and Constructs*, 1 edn, Addison-Wesley.

Silberschatz, A. og Galvin, P. B. (1994), *Operating System Concepts*, Addison-Wesley.
*http://www.bell-labs.com/topic/books/os-book/

Silberschatz, A. og Hillyer, B. K. (1996*a*), On the Modeling and Performance Characteristics of a Serpentine Tape Drive, *in* '1996 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems, Philadelphia, PA', Bell Laboratories, ACM, pp. 170–179.
*http://www.bell-labs.com/org/1123/who/~bruce/bkh_papers.html

Silberschatz, A. og Hillyer, B. K. (1996*b*), Random I/O Sheduling in Online Tertiary Storage Systems, *in* '1996 ACM SigMOD International Conference on Management of Data, Montreal, Canada', Bell Laboratories, ACM, pp. 195–204.
*http://www.bell-labs.com/org/1123/who/bruce/bkh_papers.html

Silberschatz, A. og Hillyer, B. K. (1996*c*), Storage Technology: Status, Issues, and Opportunities. submitted for publication.
*http://www.bell-labs.com/org/1123/who/bruce/bkh_papers.html

Silicon Graphics Corporation (1997), 'Xfs technical spesifications', WWW.
*http://www.sgi.com/Products/hardware/challenge/XFS/XFSspecs.html

Sony Corporation (1997), 'Product information', WWW.
*http://www.ita.sel.sony.com/products/storage/

Sørensen, G. (1996*a*), 'CGI++ – A CGI Interface object for C++'.
*http://www.pvv.ntnu.no/ gsi/software/src/cgi++/cgipp.html

Sørensen, G. (1996*b*), VRDB – Virtual Reality Database, Technical report, IDT/NTNU.

Sørensen, G. og Sandstå, O. (1996), ECM – Elvira Catalog Manager, Technical report, IDT/NTNU.

Stallings, W. (1994), *Data and computer communications*, 4 edn, MacMillan. ISBN 0-02-415441-5.

Stevens, W. R. (1990), *UNIX Network Programming*, 1 edn, Prentice Hall. ISBN 0-13-949876-1.

Stevens, W. R. (1992), *Advanced UNIX Programming in the UNIX Environment*, Professional Computing Series, Addison-Wesley. ISBN 0-201-56317-7.

Sætre, R. (1997), 'Elvira Tape Server'. Unpublished technical manual.

Sætre, R. og Sandstå, O. (1997), Video på digitale band, Technical report, IDT/NTNU.

Stroustrup, B. (1991), *The C++ Programming Language*, 2 edn, Addison Wesley. ISBN 0-201-53992-6.

Sun Microsystems (1996), 'Sun MediaCenter Server – Marketing White Paper', WWW. *http://www.sun.com/products-n-solutions/hw/servers/smc_external.html

SyQuest Technology, Inc. (1996), 'SyJet 1.3GB Removable Cartridge Hard Drive'. *http://www.syquest.com/syquest/specsj13.htm

Tandberg Data (1996), *Tandberg MLR1 Series – Reference Manual*.

Taylor, H. M. og Karlin, S. (1994), *An Introduction To Stochastic Modeling*, revised edn, Academic Press, chapter 9, pp. 485–521.

Tech Computers (1997), 'Komponenter: Harddisker', WWW. Tech is known to be a cheap retailer of hardware in Norway. *http://www.tech.no/komponenter/harddisk.html

Troll Tech (1996), *QT Documentation*. Object oriented C++ toolkit for GUIs. *http://www.troll.no/qt/index.html

Van Meter, R. og Stith, J. (1996), 'comp.arch.storage FAQ', USENET. *ftp://rtfm.mit.edu/pub/usenet-by-group/comp.arch.storage/comp.arch.storage_FAQ_1_2

Wave Technologies (1997), 'Product catalog', WWW. Wave is a Norwegian supplier of hardware. *http://www.wave.no/

Wilkinson, B. (1991), *Computer Architecture – Design and Performance*, Prentice Hall, chapter 2, pp. 41–51.

Zöckler, M. og Wunderling, R. (1996), 'class DOC++'. *http://www.ZIB-Berlin.DE/VisPar/doc++/doc++.html

Tillegg A

Ordliste

Her er en liste med forklaring på begreper, forkortelser og norske oversettelser av engelske datauttrykk som blir brukt i denne rapporten. Forklaringer tatt fra bokmålsordboken er merket med (BM).

aksess – akses's m1 (fra lat., eg 'adgang') særl i edb: adgang, mulighet til å overføre informasjon, tilgang (BM)

API – Application Program Interface, benyttes om det grensesnittet som et programsystem tilbyr for å muliggjøre interaksjon for applikasjoner som skal benytte systemet.

Applet – Et Java-program som vil bli overført til klientenes maskiner på Verdensveven for å bli utført der.

ATL – Ofte brukt forkortelse for automatiske båndbiblitoteker (automatic tape libraries).

ATM – Asynchronous Transfer Mode. Betegnelse på teknologi brukt svitsjede høykapasitetsnettverk der fiberoptikk har en sentral plass.

avlusing – Norsk ord for debugging.

barneprosess – Norsk ord for child process.

C – Programmeringsspråket størstedelen av operativsystemet UNIX er implementert i (Kernighan og Ritchie 1988).

C++ – Programmeringsspråket som størstedelen av Elvira Storage Manager er implementert i (Stroustrup 1991).

CORBA – Metode for å lage distribuerte objektorienterte programsystemer. Se kapittel 5

COTS – Commercial Off-The-Shelf, dvs. «hyllevare».

CVS – CVS er et overbygg for RCS, og tillater flere programmerer å jobbe mot de samme filene parallelt.

DOC++ – et verktøy for å automatisk generer systemdokumentasjon ut fra kommentarer i kildekode.

ECC – Error Correcting Code. Ekstra informasjon som gjør f.eks. en båndstasjon i stand til å korrigere feil som har oppstått i selve dataområdet på en blokk.

ECM – Elvira Catalog Manager. En database med informasjon om fysisk lagerstruktur i Elvira Video Server (Sørensen og Sandstå 1996).

EER-diagram – Extended ER (diagram). Som ER, men med mulighet for arving av attributter, inspirert av objektorientert tankegang.

Elvira Video Server – Et system for leveranse av video til flere samtidige brukere over et høyhastighetsnettverk. Elvira består av en ren videoserver-del og subsystemer som ECM (Sørensen og Sandstå 1996) og ESM (Sandstå et al. 1996).

ER-diagram – Entity-Relationship (diagram). Bokspråk for logisk konstruksjon av databaser. Hjørnestene i språket er entiteter (representerer «ting» i den virkelige verden), attributter (beskriver entiteter) og relasjoner mellom entiteter.

ESM – Elvira Storage Manager. Et subsystem i Elvira Video Server som skal håndtere flytting av data mellom tertiært og sekundært lager.

FCSF – First Come, First Served. Køing av jobber etter «førstemann til mølla»-prinsippet.

HDTV – High Definition TV. Høykvalitetsfjernsyn som har vært under utvikling i noen år. Det blir nok fortsatt noen år til HDTV-apprater blir allemannseie.

headerfil – En fil som beskriver grensesnittet til funksjoner (og objekter) i C (og C++).

HTML – Hyper Text Markup Language. Et «språk» for formattering av dokumentasjon som skal legges ut på Verdensveven.

innstikkmodul – En programvaremodul som kan plugges inn i et system for å utvide funksjonaliteten.

interlacing – Metode brukt i blant annet kringkasting, der annehver linje av skjermbildet tegnes opp. Først tegnes den ene halvparten av linjene opp, deretter den andre.

iterasjon – gjennomgang.

Java – Programmeringsspråk som er plattformuavhengig og svært populært i forbindelse med applikasjoner som skal ha grensesnitt mot Verdensveven.

klasse – En samling av objekter som har noe til felles.

IP – Internet Protocol. En protokoll som kort sagt får mange heterogene nett til å virke som ett stort homogent nett, der alle kan kontakte alle.

ISO – Den internasjonale standardiseringsorganisasjonen.

I/U – innputt / utputt. Fornorsking av det amerikanske uttrykket I/O (input/output). Brukes om den delen av datamaskinen som har ansvaret for kommunikasjon mellom prosessor og eksterne enheter (harddisker, nettverk, skjerm, måleutstyr, etc.)

isokron – som har samme svingetid (f eks om pendler), jf synkron (fra bokmålordboka).

JPEG – Joint Photographic Expert Group. En standardiseringskomite som har arbeidet med komprimering av stillbilder.

kjøreplan – Norsk ord for schedule.

LAN – Local Area Network. Korte ventetider og typisk store overføringshastigheter (10 - 150 Mbit/s), som ventes å øke ytterligere om kort tid.

LRU – En lagerutskiftingsalgoritme som baserer seg på at et dataområde som nylig er brukt antakeligvis vil bli brukt igjen, og at man derfor kvitter seg med områder som kun har blitt brukt for «lenge siden».

NOK – norske kroner, valutaen benyttet i Norge. Dette er en valutakode som følger ISO 4217-standarden.

NP-komplette problemer – Problemer som 50 år med algoritmeforskning ikke har klart å løse i polynomisk tid (tid av størrelsesorden N^k der k er et heltall og N er størrelsen av problemet).

objekt – En «ting» i den virkelige verden.

make – Verktøy for å bygge kjørbare filer ut fra kildekode.

migrere – migre're v2 (fra lat. 'vandre, flytte bort') om folk og dyr: vandre, flytte seg (BM)

Motif – Bibliotek for programmering av grafikk under UNIX/X.

MLR-1 – Tandbergs nye båndstasjon for QIC-bånd.

modul – i denne oppgaven brukes ordet modul om en samling kode med funksjonalitet rettet mot en spesifikk oppgave. En modul kan bestå av en eller flere klasser, og kan kjøres i samme prosess som andre moduler (serielt) eller i en egen prosess eller tråd (parallelt).

MPEG – Moving Picture Expert Group. Betegnelsen på en standardiseringskomite som arbeider med komprimering av video. Betegnelsen brukes også om standardene denne gruppen har kommet fram til.

MTBF – Mean time before failure. Mer informasjon finnes i kapittel 4.

MTTR – Mean time to repair. Forventet tid for hvor lang tid det tar å reparere en ødelagt enhet.

NASA – National Aerospace and Space Administration, USAs romfartsorganisasjon.

NFS – Network File System. Gjør det mulig å benytte harddisker på andre maskiner ved at data blir overført over internett.

NRK – Norsk Rikskringkasting. Norges største bedrift innen fjernsyn og radio.

prosess – «et kjørende program».

parametrisert klasse – Norsk ord for template-konstruksjonen i C++. Gjør det mulig å bruke en klasse som «kakeform» for å lage store mengder spesialiserte klasser.

Purify – Verktøy som analyserer C og C++-programmer for å finne minnelekkasjer og andre feil som kompilatorene ikke oppdager.

RCS – Et verktøy for administrasjon av kildekode.

RDBMS – Relational Database Management System. Et system for relasjonsdatabaser (Elmasri og Navathe 1994).

ressursfordeler – Norsk ord for scheduler.

RPC – Remote Procedure Call. Verktøy for å forenkle kommunikasjon mellom programmer. Se kapittel 5

RS232 – Grensesnitt for kommunikasjon over enkle, serielle kabler. Mye brukt for å koble modemer til datamaskiner, osv.

socket – Grensesnitt mot TCP/IP og UDP/IP under UNIX (og også Windows). Se beskrivelse i kapittel 5.

Solaris – En versjon av SunOS.

STL – Standard Template Library. Et standard klassebibliotek som vil bli en del av den nye ANSI/ISO C++-standarden.

SunOS – Sun Microsystems versjon av UNIX.

surround-lyd – lyd som oppfattes tredimensjonalt, i motsetning til vanlig stereolyd som er todimensjonal.

striping – Spredning av data over flere lagringsmedier. Det står mer om striping i kapittel 1.

størrelsesorden – annet ord for 10^x. 100 er av størrelsesorden 2, mens 100 000 er av størrelsesorden 5.

synopsis – (gr 'overblikk') oversiktlige sammenstillinger, parallelloppstillinger av et emne (BM).

TCP – Transfer Control Protocol. En protokoll som ligger oppå IP og som tilbyr forbindelsesorientert kommunikasjon mellom to applikasjoner koplet til Internett.

tegnstrøm – Norsk ord for bytestream.

tråd – I operativsystemsammenheng er en tråd en «lettvektsprosess» som kjører i samme minneområde som andre tråder. Tråder gjør det mulig å ha høy grad av parallellitet med mindre kostnader enn for multiprosessering, der alle prosessene har sine egne minneområder.

UDP – Unreliable Datagram Protocol. Protokoll for forbindelsesløs kommunikasjon over IP.

UNIX – Operativsystem mye benyttet i akademiske miljøer og i bedrifter med behov for store distribuerte maskinsystemer.

USENET – Ofte kalt News. Et debattsystem flittig brukt av internettbrukere.

Verdensveven (World Wide Web, WWW) – Et verdensomspennende informasjonsnettverk, basert på klient/tjener-teknologi og Internett.

WAN – Wide Area Network. Nettverk som knytter sammen klienter eller andre nettverk over forholdsvis store geografiske avstander. Et WAN vil alltid ha høyere ventetider enn et LAN på grunn av den geografiske avstanden (signaler beveger seg med en hastighet på mellom $2 \cdot 10^8$ og $3 \cdot 10^8$ meter per sekund).

Web – Se verdensveven.

WWW – Se verdensveven.

X – UNIX kjører en program kalt X på hver maskin som vil kjøre grafiske applikasjoner. Applikasjoner kobler seg opp til X (som egentlig er en tjener) når de vil vise noe på skjermen.

X.25 – Betegnelse på nettverk som blir tilbudt mange store telekommunikasjonsselskaper.

Tillegg B

Møte med NRK

Sammen med veileder og faglærer har undertegnede deltatt på to møter med NRK-ansatte, der vi har fått vite mer om NRKs eksisterende arkiv og fremtidige planer på dette området. Før møtet satte vi opp endel punkter som vi ville ta opp på møtet:

- Informasjon om dagens NRK-arkiv
  - Hva er den omtrentelige lagringsmengde?
  - Hvilken rate øker arkivmengden med?
  - Hvilke typer materiale har man?
  - Hva slags kvalitet det er på dette materialet?
  - Hvem bruker arkivet i dag?
  - Finnes det noen spesielle bruksmønstere?
  - Hvor mange forespørsler kommer det mot arkivet i dag?
  - Hvordan fungerer arkivet i dag?
  - Hvor godt utstyrt er NRKs ansatte (de som vil bruke arkivet)?
- Morgendagens NRK-arkiv
  - Hvor stor del av det eksisterende arkivet kan NRK tenke seg å digitalisere ?
  - Hva slags video-kvalitet bør arkivet ha?
  - Hva synes NRK om forslaget til køing av forespørsler?
  - Finnes det planer for utbygging infrastruktur på datasiden?

B.1 Møtereferat

Dato:	Fredag 1996-12-13
Tid:	10:00-12:30
Tema:	*Møte mellom IDT og NRK om videoarkivtjenere og fremtidig arkivløsninger.*
Sirkulering:	Geir Sørensen, Olav Sandstå, Roger Midtstraum, Rune Sætre. Alle IDT.
Kvalitet:	Løse setninger. Notataktig.
Annet:	Anmerkninger står i klammer.

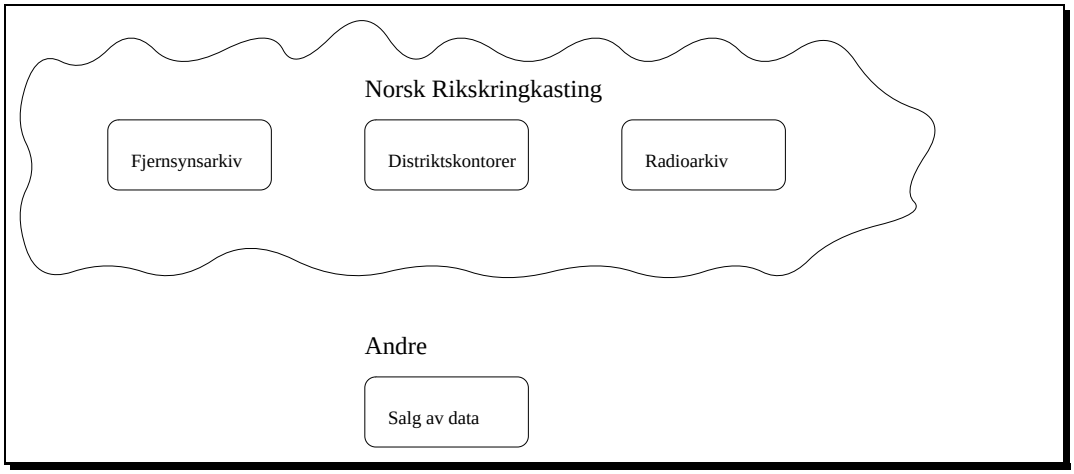
Oppmøte og presentasjon

Deltagelse:

- Roger Midtstraum, Amanuensis IDT
- Geir Sørensen, Diplomstudent IDT
- Olav Sandstå, Stipendiat IDT (forfall)
- Tor Øye, NRK: Prosjektleder for forprosjektet for nytt arkiv i NRK. Vært i NRK siden 1962.
- Arild Bakken, NRK: Kollega av Tor Øye. Også vært i NRK siden 1962.
- Oddbjørn Holgersen, NRK: Jobber i den operative avdelingen. Bearbeidingsseksjonen. Redigerer programmer. Utdannet fotograf og klipper. Undervist i kildekritikk på TV-siden.
- Siri Undall, NRK (forfall)
- Ole Petter Olsen møtte istedet for Undall. Jobber i samme avdeling som Bjørn Klock. Vært innom arkivet tidligere. Jobber med Prince (produksjonsplanlegging)
- May Noreld: Fjernsynsarkivar. Jobbet i NRK siden 1964 (samme sted). Jobbet mye med SIFT.
- Tedd Johansen, NRK (forfall)
- Karin Aall Schaubert, NRK: Arkivbruker. Programsekretær og bibliotekar. Brukerrepresentant i forprosjektgruppen.
- Bjørn Klock, Overingeniør i tele og dataavdelingen NRK. Prosjektleder for NRKs del av LAVA-prosjektet.

Om forprosjektet til NRK på arkiv

Utredning om innsatsområder for et nytt arkiv. Man analyserer volumbehov, formater og rutiner. Digitalisering og søking.



Figur B.1: Arkiver i NRK

I dag er det fryktelig mange formater i NRKs arkiv. 2', 1' bånd. Film. Betamax, digital beta, etc. *Mange formater fører til dårlig tilgjengelighet.*

NRKs arkiv mangler f.eks. mulighet for å vise 2' og 1' bånd. Dette er svære maskiner som det kun er noen få igjen av i NRK. Det pågår arbeid med å spille materiale fra 2' og 1' over til digitale bånd.

En del spørsmål som ble behandlet i forstudiet:

- Digitalisering: hva er fullverdig kvalitet?
- Volum: ca. 90 000 timer materiale i fj.ark.
- Tilgjengelighet: manuell uthenting, etc.
- Sikring: hvor sikkert er materialet lagret.

Svar fra NRK

Et fremtidig videoarkivs volum

Det dreier seg stort sett om nyheter, sport og dokumentarisk stoff. Dette utgjør omtrent halvparten av mengden i arkivet. Man skal lagre reportasjer, ikke alt som sendes (introer, plansjer, etc.).

Resident på harddisker: ca. 1 mnd vil være aktuelt.

En jukebox med rimelig aktuelt stoff? Resten manuelt?

Kun aktuelt med et system for lagring av materiale i påsynskvalitet[1]

Kvalitet

NRK-folkene synes kvaliteten på det de så på folkemuseet var godt nok. Det er aktuelt med forskjellige rater for forskjellige typer innslag/program. Tenk f.eks. på forskjellene mellom et turninnslag og et tv-teater.

Synkronisitet og realtime overføring er viktigere enn at rammene ankommer feilfritt. (Feilkorrigering er underordnet realtime-kravene). [Dette gjelder produksjonsprosess, antar jeg. GS]

Endel av NRKs sendinger sendes med utstyr som gjør at man får tap av kvalitet. Dette egner seg ikke for gjenbruk. (Dette gjelder f.eks. produksjoner fra Medieoperatørene på NRK-TO)

Beskrivelsesprosessen og dagens arkiv

Bruksrettigheter

Trafikklysmodell foreslått for bruksrettigheter:

- Grønn: kan benyttes uten å kontakte rettighetshaver, eller at det ikke finnes noen rettighetshaver.
- Gul: rettighetshaver må godkjenne bruk.
- Rød: kan ikke benyttes uansett. (Gradert materiale?)

Dagens arkiv

Søkefunksjoner for NRKs brukere i dagens dokumentarkiv (SIFT):

- Fritekstsøk (på alle typer metadata)
- Årstider (vanskelig med dagens system, må søke på prod.dato)
    - vær og føre
    - vinterbilder av Sogn og Fjordane

[1] Påsynskvalitet: lavere kvalitet som man bruker for å se igjennom materialet på når man forbereder en produksjon. Produksjonskvalitet: høy kvalitet

- Emneord
    - personer
    - steder
    - områder

Dagens metode for å finne materiale: Søk i SIFT → treff → påsyn (video)

Hver 2. time leverer arkivarene materiale til påsynsrommene. Det finnes også andre NRK-ansatte som får lov til å hente ut materiale direkte fra arkivet. Dette skjer mye på kveldstid, når arkivarene har gått hjem. Arkivet er egentlig bare åpent fra ca. 8-16. Det er Dagsrevyen og Sportsrevyen som stort sett bruker arkivet om kvelden.

2/3 av brukerne ber om aktuelt stoff.

Det er stor forutsigbarhet for henting av data på endel områder. Særlig om ting som går igjen periodevis. Dette gjelder f.eks. utdeling av Nobels fredspris, statsbudsjettet, etc.

Antall brukere i dagens arkiv:

- 3-4 arkivarer
- 2 stykker fra hver avdeling i NRK har også tilgang
- tilsammen 15-20 mennesker som bruker arkivet.

160 000 registrerte dokumenter i SIFT. Se vedlegg.

NRK har fått til et webgrensesnitt mot dagens SIFT. WinSIFT er oppgitt av Statens datasentral.

Søk i en fremtidig videodatabase

Hvor lang bør den snutten man henter ut være:

- minsteenhet: sekvens
- største: hele programmet (eller innslaget i en dagsrevy)

Hvordan skal man avgrense snuttene? Det finnes ingen støtte for dette i dag. Det er planlagt at man skal begynne å lagre TC-er i selve bildedataene (det finnes rom for lagring av data i pal-formatet).

Det må være mulig å låse visse data i diskbufferne.

Thumbnail-montasjer som viser første ramme i hver sekvens?

Det må være samsvar mellom det som vises i påsynskvalitet og originalmaterialet. Ting som ikke kan hentes fram i originalkvalitet må avmerkes. [Dette er viktig. GS]

Det er i dag en stor mengde bånd som hentes ut unødvendig, fordi man ikke helt vet hva man er ute etter. Dette kan avhjelpes betrakelig med en videoarkivtjener.

Arkivets fremtid

Man ser en økning av bruken av arkivet i fremtiden.

Lagret materiale per år øker jevnlig (lineært?). De siste seks åra har det økt med ca. 1000 ekstra dokumenter i året. Dokumentene kan ha varierende lengde, så dette sier ikke noe om lagret video. Innføringen av NRK2 fører til ytterligere økning (kvantesprang).

Tillegg C

Beregninger og illustrasjoner

C.1 Lagringsmengder

C.1.1 Total lagringsmengde av videodata

365 dager med en halvtimes Dagsrevy-sendinger vil utgjøre en betydelig datamengde. Dersom man har lagret dette med f.eks. MPEG-1 (se avsnitt 3.3.2), vil man få $365 \cdot 0.5 \cdot 1.8 \cdot 10^6 = 3.28 \cdot 10^8 bit = 41GB$ data. La oss si at man vil ha tilgang til 4 timer produsert materiale per dag, fra de siste 30 år, så får man $30 \cdot 8 \cdot 41GB = 10TB$ materiale man skal ha tilgang til. Med en pris på 1 krone per MB vil det koste 10 millioner kroner å ha alt på harddisk, noe som er ganske utenkelig.

C.1.2 Lagringsmengde av indekser

Dersom man for en halvtimes dagsrevy-sending velger å indeksere hver eneste ramme, vil antall rammer i indeksen være: $n_{rammer} = 1800s \cdot 25s^{-1} = 45000$. Dersom hver ramme tar 24 byte (12 byte for rammenummer og 12 byte for blokkadresse), får vi $K_{indeks}(1) = 45000 \cdot 12byte = 1.08MB$. 365 dager med Dagsrevyer vil da føre til en diskforbruk på: $K_{indeks}(365) \simeq 39MB$.

39MB er kanskje ikke en avskrekkende størrelse, men med tanke på at man i tillegg må ha like store indekser for to lydkanaler, og kanskje enda fler indekser for alternative videoformater og simultane lydspor (f.eks. dubbing), så kommer man raskt opp i over hundre MB for lagring av Dagsrevyer over et helt år. I tillegg blir det produsert annet stoff (f.eks. 9.5 timer per dag) som også skal lagres med indekser, og da vil man kanskje komme opp i rundt en gigabyte lagring for indekser hvert år. Etter noen år vil man derfor få ganske uhåndterlige størrelser på indeksdatabasen dersom den kun skal befinne seg på disk.

C.2 Brukere av VOD versus videoarkivtjenere

En typisk VOD-tjener har kanskje 500 GB med filmer lagret på harddisk og fem tusen brukere[1] (br). Dette koster altså:

$$K_{VOD} = p_{HD} \cdot 500GB = 0.625MNOK$$

$$p_{bruker} = \frac{K_{VOD}}{5000br} = 125NOK/br$$

Dersom vi antar at (et ganske populært) videoarkiv har 200 brukere, får vi:

$$K_{arkiv} = p_{HD} \cdot 500GB = 0.625MNOK$$

[1] Ikke samtidige, men totalt antall brukere som har tilgang til tjeneren!

$$p_{bruker} = \frac{K_{ark}}{200br} = 3.13kNOK/br$$

Dette er riktignok tall tatt ut fra løse luften, men det gir et visst inntrykk av forholdene likevel. Dersom man skal drive et videoarkiv med kun harddisklagring, vil det koste en til to størrelsesordner mer per bruker. (Dessuten vil arkiver gjerne bli mye større enn 500GB. NRK har f.eks. rundt 90000 *kassetter* med videomateriale). Dersom man antar at hver kassett representerer 10 GB, har man et arkiv på rundt 900TB!

C.3 Båndbredde i arkiver og i VOD

Dersom et VOD har et 500GB RAID med 50 disker à 10GB, der hver disk har en netto gjennomsnittlig overføringsrate på 10MB/s (inklusive overhead for operasjoner som RAID-programvaren/maskinvaren må gjøre på hver datablokk) får vi 500MB/s akkumulert overføringsrate. Dette gir følgende tall for et diskbasert system

$$R_{VOD} = \frac{500MB/s}{2000} = 250(kB/s)/br$$

$$T_{VOD} = \frac{500GB}{500MB/s} = 0.3h$$

$$S_{VOD} = \frac{500MB/s}{2.0Mbit/s} = 2000$$

Verdien for antall strømmer (S) sier at vi kan betjene alle kundene samtidig (100% dekningsgrad).

Dersom arkivet vårt på 500GB kun har en båndstasjon med en overføringsrate på 1.5 MB/s, har vi følgende verdi for overføringsrate per bruker (R) og tid det tar å lese hele arkivet (T):

$$R_{arkiv} = \frac{1.5MB/s}{200} = 7.5kB/s$$

$$T_{arkiv} = \frac{500GB}{1.5MB/s} = 92.6h$$

$$S_{arkiv} = \frac{1.5MB/s}{2.0Mbit/s} = 6$$

Dette sier at vi bare kan betjene 6 av 200 brukere samtidig, altså en dekningsgrad på bare 3% over tid. Dette er et såpass lavt tall at man kan snakke om at båndstasjonen utgjør en flaskehals i arkivet når det gjelder båndbredde.

C.4 Pålitelighet i parallelle og serielle systemer

I dette avsnittet er det kort vist hvordan man kommer frem til uttrykkene for pålitelighet for parallelle og serielle systemer, der hver komponent har en egen pålitelighetsfunksjon.

For serielle systemer har vi at:

$$\begin{aligned} R(n,t) &= P(T > t) \\ &= P(T_1 \geq t, T_2 \geq t, \ldots, T_n \geq t) \\ &= P(T_1 \geq t) \cdot P(T_2 \geq t) \cdots P(T_n \geq t) \\ &= R_1(t) \cdot R_2(t) \cdots R_n(t) \\ &= \prod_{i=1}^{n} R_i(t) \\ &= R(t)^n \end{aligned} \tag{C.1}$$

For parallelle systemer har vi:

$$\begin{aligned} R(n,t) &= P(T > t) \\ &= 1 - P(T \leq t) \\ &= 1 - P(T_1 \leq t, T_2 \leq t, \ldots, T_n \leq t) \\ &= 1 - P(T_1 \leq t) \cdot P(T_2 \leq t) \cdots P(T_n \leq t) \\ &= 1 - (1 - R_1(t)) \cdot (1 - R_2(t)) \cdots (1 - R_n(t)) \\ &= 1 - \prod_{i=1}^{n} (1 - R_i(t)) \\ &= 1 - (1 - R(t))^n \end{aligned} \tag{C.2}$$

Tillegg D

Detaljert systemdokumentasjon

D.1 Systemfiler

D.1.1 Utkast til protokollspesifikasjon

```
------------------------------------------------------------------------
-- ELVIRA STORAGE MANAGER PROTOCOL
--
-- $Id: protocol.txt,v 1.1 1997/03/11 20:06:32 gsi Exp $
--
-- Initiated by: Geir Sørensen
-- Last updated by: same
------------------------------------------------------------------------

------------------------------------------------------------------------
--Request to server:
------------------------------------------------------------------------

REQUEST_FROM_CLIENT:
  "client_req"           -- message ID
  INIT_REQ | OP_REQ | END_REQ
  "end"

INIT_REQ:
  "init_req"             -- message ID
  (integer priority)     -- for future use
  (string user)          -- user ID
  (string passwd)        -- string with password (encrypted)
  (bool debug)           -- debugging?
  (string debugmail)     -- e-mail-address

OP_REQ:
  "data_req"             -- message ID


END_REQ:
  "end_req"              -- message ID

OPERATION: -- folded into OP_REQ
  "get_videostream_info"
  (integer key)          -- get status on particular archive

| "acquire_videostream" -- ask ESM to put a whole archive into the buffer
  (integer vs_key)
  USERJOB

| "use_videostream"     -- mark replicate as used (for LRU)
  (integer vs_key)

| "release_videostream"
  (integer vs_key)

| "lock_videostream"
  (integer vs_key)

| "unlock_videostream"
  (integer vs_key)
```

```
OP_REQ:
  OPERATION

USERJOB:
  -- job specified as in class ESM_UserJob (esm/jobs.h)

-----------------------------------------------------------------------
Response from Server:
-----------------------------------------------------------------------

RESPONSE_FROM_SERVER:
  "server_resp"
  INIT_RESP | OP_RESP | END_RESP | ERROR

INIT_RESP
  "init_resp"
  INIT_REQ

OP_RESP:
  "data_resp"

END_RESP:
  "end_resp"

ERROR:
  "error_resp"
  (integer nerror)              -- number of lines of errors
  (string errmsg)


SYSTEM_ERROR:
  "system_error"
  (integer nerr)
  (string errmsg)



-------------------------------------------------------------
--PDU
-------------------------------------------------------------

PROTOCOL:
(string version)                -- currently "ESM_PROT_1.0"
REQUEST_FROM_CLIENT | RESPONSE_FROM_SERVER
"end"
```

D.2 Detaljerte klassebeskrivelser

På neste side følger en lang rekke sider med detljert dokumentasjon av klassenes grensesnitt. Dokumentasjonen er generert med verktøyet DOC++ (Zöckler og Wunderling 1996) ut fra alle headerfilene i esm- og util-katalogen (nevnt i kapittel 7). Dersom leseren har adgang til Verdensveven, anbefales det å heller benytte seg av HTML-versjonen som ligger på undertegnedes private hjemmeside `http://www.pvv.org/~gsi/projects/thesis/`.