Video on Demand med Digital Rights Management og publisering av innhold for VoD tjenester.

av

Espen Westlye Stråpa

Hovedoppgave til mastergraden i
informasjons- og kommunikasjonsteknologi

Høgskolen i Agder

Sammendrag

Oppgaven ser nærmere på hvordan en kan beskytte digitaltinnhold for en Video on Demand tjeneste med Digital Rights Management og hvordan en enkelt kan publisere innhold for en slik tjeneste. Det blir skissert løsninger for å automatisere publiseringen av innhold. Andre aspekter ved oppgaven er hvilke logging funksjonalitet som finnes i streamingserveren og fordeler/ulemper ved ulike distribusjonsalternativer for digitaltinnhold over Internett. Distribusjonsalternativene som testes er ulike streaming og nedlastingsalternativer med hensyn på oppnåelig videokvalitet, brukervennlighet og fra aksessleverandørens ståsted med en nettverksøkonomisk synsvinkel.

Oppgaven ser på hvordan innholdsleverandører og aksessleverandører spiller sammen for å publisere innhold for en VoD tjeneste som benytter DRM. Det omhandler blant annet hvordan en beskytter innhold, distribuerer lisenser og distribuerer innholdet til sluttbrukerne. En sentral del er hvordan en kan automatisere publiseringsprosessen av innhold. Oppgaven beskiver ulike systemoppsett og komponenter for å automatisere publiseringen av innholdet og det er laget prototyp systemer som viser at arkitekturen funger.

DRM teknologien fra Microsoft er fleksibel og enkel å integrere i nye eller eksisterende systemer. Ved bruk av DRM kan en enkelt administrere tilgangen til innholdet ved hjelp av lisenser. Innholdet beskyttes ved at innholdet krypteres og brukeren trenger en lisens for å kunne benytte innholdet.

Den automatiserte publiseringsprosessen som er skissert i oppgaven vil minimere arbeidet hos innholdsleverandør og aksessleverandør for å publisere innhold. Arbeidet viser at en må benytte DRM for å beskytte digitalt innhold for både streaming og nedlasting av innholdet. Distribusjonsalternativet som fungerer best for en VoD løsning er streaming med fast streaming funksjonalitet. Progressiv nedlasting er et bra alternativ dersom en ønsker å fokusere på videokvaliteten, men har noen negative sider ved brukervennligheten som gjør at det ikke er det beste alternativet for distribusjon av innhold. Progressiv nedlasting kan tilbys som en tilleggstjeneste til streaming med fast streaming funksjonalitet, men ikke som en enkelt stående løsning.

Forord

Diplomoppgaven ble skrevet ved Høgskolen i Agder som en del av min Master i teknologi grad ved høgskolen. Den ble utført i samarbeid med Telenor iCanal og arbeidet ble gjennomført i perioden mellom Januar 2003 og May 2003.

Det kreves basiskunnskaper innenfor streamingteknologi, Microsoft.NET programmering og databasehåndtering for full forståelse av rapporten, men rapporten gir også en innføring i prinsippene for Digital Rights Management for Video on Demand systemer.

Jeg vil takke min oppdragsgiver Telenor iCanal for veiledning og lån av utstyr slik at oppgaven kunne gjennomføres. Jeg vil også takke min veileder Ola Torkild Aas for god hjelp gjennom arbeidet med oppgaven.

Grimstad, mai 2003

Espen Westlye Stråpa

Innholdsfortegnelse

Figurliste

Tabelliste

1 Innledning

1.1 Bakgrunn

Video on Demand er et aktuelt tema for tiden ettersom bredbåndsdekkingen og nedlastingshastigheten øker i volum. Utviklingen baner vei for nye og spennende distribusjonskanaler for innhold over Internett. Innholdet som distribueres kan tilpasses hver enkelt sluttbruker noe som skiller seg klart fra de tradisjonelle mediene. Brukeren kan i dag ikke bestemme når han ønsker å se et spesielt program eller film på TV noe som er fastsatt av TV selskapenes sendeskjema. En rekke kabel og satellitt selskaper har i dag tjenester hvor brukeren kan bestille filmer til fastsatte klokkeslett. Men denne løsningen har begrensninger som for eksempel at filmene starter til fastsatte klokkeslett og utvalget av filmer begrenses av antall kanaler som er avsatt til dette formålet.

Nedlastingshastigheten for bredbåndsabonnementer øker noe som øker videokvaliteten en kan levere over bredbånd, men også en annen faktor spiller inn på videokvaliteten. Nye komprimeringsalgoritmer gjør at en kan levere innhold i lavere bitrater med bedre video og audio kvalitet enn tidligere etter hvert som komprimeringsteknikkene utvikles. For eksempel har Windows Media 9 formatet en forbedring på 15 til 50 prosent i forhold til Windows Media 8 kodeken når det gjelder videokvalitet/båndbredde. Kombinasjonen av økt nedlastingshastighet og bedre komprimeringsteknikker gjør bredbånd stadig mer attraktivt for distribusjon av digitalt innhold.

Distribusjon av innhold over bredbånd vil gi brukere mye bedre interaktivitet når det gjelder bestilling av innhold over nettet. Brukere får tilgang til det innholdet de ønsker når de måtte ønske det. Utvalget av filmer begrenses av lagringskapasitet på sentrale server og båndbredde ut fra disse. For at brukere skal kunne benytte en slik tjeneste må han være tilknyttet en bredbåndsleverandør og gjerne ha en PC med TV utgang slik at innholdet kan vises på TV’en. Noe som gjør dette enda mer spennende er at det i dag utvikles setupbokser som kan kobles direkte til TV'en. Setupboksene fungerer som bindeleddet mellom TV din og Internett, de tar altså over oppgavene den tradisjonelle PC'en har i dag. Flere og flere DVD spillere vil sannsynligvis ha denne funksjonaliteten innebygd som standard om kort tid. Blant annet har KISS allerede laget en DVD spiller som kan fungere som en setupboks. Den har nettverkstilkobling innebygd som en plugger rett til bredbåndlinjen.

Distribusjon av innhold over Internett fører til at en må ta i bruk nye metoder for å beskytte rettighetsbelagt materiale. En må ha en sikker måte å distribuere innholdet

slik at rettighetene som er knyttet til innholdet blir overholdt. Tradisjonell rettighetsbeskyttelse er knyttet til fysiske medier som for eksempel DVD plater, CD plater eller bøker. Hvis en kjøper en DVD, CD eller bok så følger selvfølgelig bruksretten til produktet med. Som alle vet så er det ulovlig å kopiere denne typen materiale, det vil oppfattes som piratkopiering. I dagens Internett hverdag så er problemene med piratkopiering av innhold store og for å få med store innholdsleverandører må en ha distribusjonsmetoder som kan sikre innholdet mot piratkopiering. Digital Rights Management, heretter forkortet til DRM, brukes til å sikre innholdet mot misbruk. DRM går kort sagt utpå at innholdet krypteres noe som gjør det ubrukelig for alle som ikke har gyldig lisens til å benytte innholdet. Lisensen fungerer som en nøkkel for å låse opp innholdet slik at det kan benyttes og administrer på denne måten tilgangen til innholdet. Hvordan DRM fungerer mer i detalj finner en i avsnitt 1.1.1 og kapittel 2.

Distribusjonen av innhold kan deles inn i ”on Demand” tjenester og ”live broadcast” tjenester. For begge typer kan en distribuere innholdet enten ved unicast eller multicast. Siden Telenors bredbåndsnett ikke støtter multicasting per i dag så er dette arbeidet konsentrert rundt unicast. Oppgaven vil se nærmere på en videokiosk tjeneste hvor brukere kan leie filmer over Internett. Distribusjonen av innhold kan gjøres etter to modeller enten ved streaming eller nedlasting av innholdet. Modellene har ulike positive og negative sider, streaming fører til at videokvaliteten på innholdet begrenses av båndbredden til brukeren. Dersom en ønsker å fokusere på videokvalitet og tilby kvalitet som tilsvarer for eksempel DVD må en ha tilgang til høyere båndbredder enn det som er vanlig for dagens ADSL abonnementer. En kan kompensere for manglende båndbredde ved å laste ned innholdet til brukeren før han setter i gang filmen, noe som vil gi en lang bufferingtid for brukeren før han kan starte filmen.

I denne oppgaven er det benyttet DRM fra Microsoft, DRM produktet består av et COM objekt som ikke er noen ”stand-alone” applikasjon. Oppgaven vil se nærmere på hvordan en kan sette opp et system som benytter denne komponenten for å pakke og generere lisenser til innhold.

1.1.1 Digital Rights Management

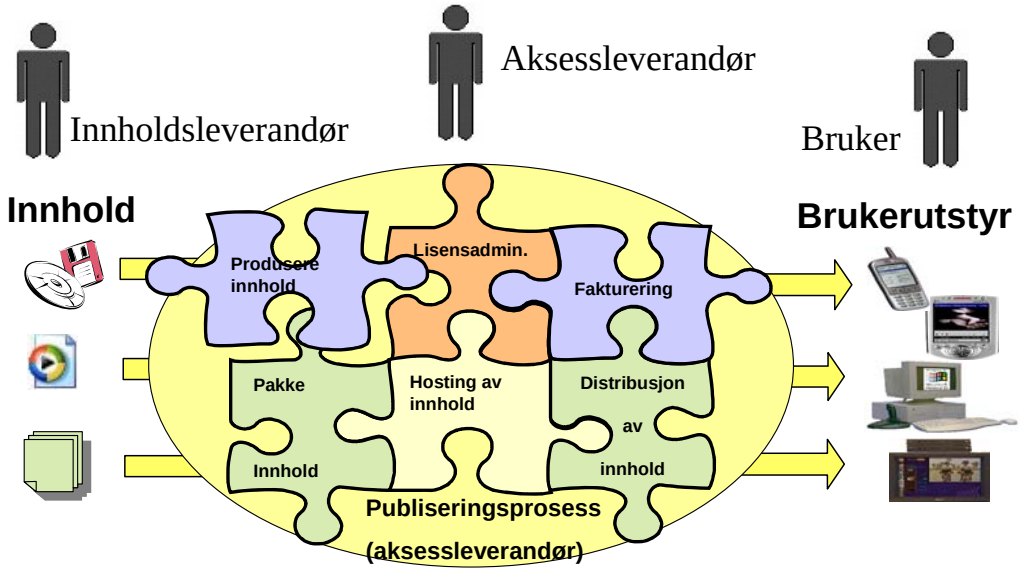
DRM består av et sett med teknologier som innholdsleverandører kan benytte til å beskytte rettighetsbelagt materiale. Det er et system som krypterer digitalt innhold og begrenser aksessen til innholdet til kun de personene som har skaffet seg en gyldig lisens for innholdet de ønsker å benytte. Microsoft sier følgende om DRM teknologien:

> *"DRM is a technology that enables the secure distribution, promotion, and sale of digital media content on the Internet.[1]"*

I et e-handelssystem for innhold er DRM et av hovedelementene. Ved å kryptere innholdsfilene kan en beskytte rettighetsbelagt innhold mot piratkopiering på en tilfredstillende måte. Det finnes mange aktører på markedet som tilbyr DRM produkter, blant annet Microsoft[2], RealNetworks[3], DMDSecure[4] og Adobe[5]. DRM brukes ikke bare for musikk og video filer, Adobe tilbyr DRM beskyttelse på sine filformater og beskriver sitt DRM system slik:

> *"Det åpner for en lang rekke muligheter," sier David Lehr, utviklingssjef i Adobes ePaper Solutions-gruppe. "For eksempel vil vi se 'superdistribusjonssmodeller' der innhold blir distribuert i stor målestokk i en organisasjon og blir betalt for basert på hvor mange som virkelig bruker det."[5]*

Et e-handelssystem for innhold består av flere aktører. En kan grovt dele inn i innholdsleverandører, aksessleverandører og sluttbrukere. Ønsker en å publisere innhold er det en del oppgaver som må deles mellom de ulike aktørene. Figur 2 viser en skisse over oppgaver som til sammen utgjør en publiseringsprosess hvor innholdet blir beskyttet med DRM. Som en kan se av figuren under består prosessen blant annet av produksjon, pakking , lisensadministrasjon, hosting, distribusjon og fakturering av innhold. Publiseringsprosessen gjør innholdet fra innholdsleverandører tilgjengelige for sluttbrukerne.



Figur 1: Oversikt over aktører i publiseringsprosessen.

Innholdsleverandørene er de som produserer innhold og eier rettighetene til innholdet, men de trenger en distributør som kan levere innholdet til kundene. I figur 1 ser en at aksessleverandøren vil ha en distributør rolle innen et Video on Demand system, de vil sørge for hosting og distribusjon av innhold til sluttbrukerne. Andre oppgaver som vil ligge hos aksessleverandør er pakking(kryptering) av innhold, betalingsløsninger og utlevering av lisenser. Det kan tenkes at enkelte innholdsleverandører ønsker å gjøre pakkingen av innholdet selv, noe som gjør at misbruk av innholdet ikke kan skje i distributørleddet. I dette tilfellet vil aksessleverandør sørge for distribusjon og utdeling av lisenser.

1.1.2 Publisering av innhold

For at Video on Demand systemer med DRM skal fungere bra i en kommersiell setting må det være enkelt å publisere nytt innhold. Det må være enkelt for innholdsleverandører å publisere sitt innhold hos en distributør. For å gjøre publiseringsprosessen enklere vil vi denne rapporten blant annet se på noen automatiserte publisering prosedyrer som gjør publiseringen enkel for både innholdsleverandør og distributør. Dersom en publiserer lite nytt innhold kan en klare seg med manuell pakking av innholdet, mens en derimot mottar flere tusen nye filer hver uke som skal pakkes og publiseres vil en automatisert publisering være veldig tidsbesparende for den som pakker og publiserer innholdet.

Sett fra en innholdsleverandør sin side så ønsker han en prosess som er lite tidkrevende og som beskytter innholdet tilfredstillende mot misbruk som piratkopiering. Innholdsleverandører og distributører inngår avtaler hvor det spesifiseres hvilke prismodeller som skal brukes for innholdet. En kan operere med forskjellige prismodeller. Et selskap kan basere seg på Pay-Per-View betaling for innholdet sitt, andre baserer seg på abonnementer på innholdet. Forskjellige prismodeller krever forskjellig lisenshåndtering. Dersom en har en abonnementsmodell vil det være lite hensiktsmessig for brukeren og lisensutgiveren at brukeren må laste ned en ny lisens for hver eneste fil brukeren ønsker å benytte. Det mest hensiktsmessige vil da være at brukeren laster ned en lisens som omfatter alle filene i et abonnement. Opererer en med en Pay-Per-View prismodell er det viktig at lisensen som lastes ned kun kan brukes på en fil. Dvs. at en trenger en lisens for hver fil en ønsker å benytte. Lisensene kan da ikke misbrukes slik at en bruker kan benytte innhold han ikke har betalt for. Oppgaven skisserer løsninger for begge disse prismodellene.

Erfaringer fra tidligere prosjekter hos Telenor iCanal, blant annet Games on Demand prosjektet, har vist at det må være enkelt for brukeren å sette opp PC'en for bruk av forskjellige tjenester. Dersom et VoD system skal fungere optimalt er det en del programvare som må settes opp og konfigureres på klientens PC. En må

ha player, kodeker, bredbåndsforbindelse og individualisert player hvis dette er et krav. Noe av dette kan være vanskelig å skjønne for uerfarne brukere så tidligere erfaringer tilsier at en må gjøre denne prosessen enkel for brukeren. Implementasjons kapittelet beskriver hva som må sjekkes på klient PC'en og hvordan en kan hjelpe brukere til å sette opp PC'en sin for VoD med DRM tjenester.

Informasjonsinnhenting kan være nytting i et VoD system. En vil i et slik system selvfølgelig ha kjøpslogger for blant annet å betalte utbytte til innholdsleverandørene og fakturering av kundene. Basert på kjøpsloggene kan en også hente inn informasjon om for eksempel de mest populære filene osv. En kan hente inn mye mer nyttig informasjon som et supplement til kjøpsloggene dersom en benytter den nye logging funksjonaliteten som finnes i Windows Media Services 9 Series. Logging funksjonaliteten som finnes i streamingserveren kan brukes til å hente inn ekstra informasjon. En kan for eksempel hente inn hvilke hastighet en film ble stream'et, hvilke klientapplikasjon som brukes, forskjellig informasjon om klient PC'ens software og hardware. Informasjonen kan være nytting grunnlag i strategiske valg når det gjelder forskjellige kodingsalternativer eller markedsføringstiltak.

1.2 Oppgavedefinisjon

Sammen med Telenor iCanal ønsket jeg å se nærmere på hvordan en kan implementere en Video on Demand løsning basert på Digital Rights Management produkter levert av Microsoft. Vi ønsket å se på hvordan en kan beskytte innhold ved hjelp av Windows Media Rights Management SDK og hvordan en effektivt og enkelt kan publisere innhold for en slik tjeneste. Vi ønsket også å se på ulike former for distribusjon av innholdet som ulike streamingalternativer og nedlastingsalternativer. Et samarbeid mellom innholdsleverandører og aksessleverandører forutsetter at det finnes en del loggig funksjonalitet i løsningen som blant annet kan benyttes til analyse av historiske trender. Vi ønsket derfor å se hvilke muligheter som finnes i streamingserveren når det gjelder logging.

Oppgave definisjonen ble som følger:
I oppgaven vil noen elementer som er viktige for en kommersiell Video on Demand tjeneste studeres.
Viktige elementer for en VoD tjeneste:

- Digital Rights Management for beskytting av innhold.
- Publisering av innhold for VoD tjenester. Herunder hvordan nettverksleverandører og innholdsleverandører spiller sammen.

- Fordeler og ulemper ved henholdsvis streaming og progressiv nedlasting av innhold.

Oppgaven vil ta for seg hvordan en benytter Digital Rights Management for en Video on Demand tjeneste på en måte som gjør publisering av innhold enkelt. Det vil si hvordan nettverksleverandører og innholdsleverandører spiller sammen for å lage en VoD tjeneste basert på DRM. VoD tjenester kan distribuere innhold på forskjellige måter enten som streaming eller som progressiv nedlasting av innhold. Hvilke fordeler og ulemper vil de ulike alternativene gi sluttbruker og hvordan fungerer de ut ifra en "nettverksøkonomisk" synsvinkel. Nedlasting vil si at brukeren fysisk laster ned filen til PC'en sin, mens for streaming så vil en motta en videostrøm som kun behandles i datamaskinens minne. En stor del av dagens ADSL linjer er ikke i stand til å levere høye nok nedlastingshastigheter slik at kvaliteten på streaming blir bra. Kan dette dekkes inn med progressiv nedlasting av filen?

Oppgavens kjerneområder er:

- Finne ut hvordan DRM beskyttelse av innhold kan implementeres
- Se på forskjellige publiseringsalternativer for innhold med DRM.
- Implementere en enkel prototyp av en VoD tjeneste som benytter DRM.
- Diskutere og konkludere på grunnlag av resultater.

Tittelen på diplomoppgaven ble som følger:

> *Video on Demand med Digital Rights Management og publisering av innhold for VoD tjenester.*

1.3 Litteraturstudie

Avsnittet beskriver hvor en kan finne informasjon om Video on Demand og Digital Rights Management systemer. I oppgaven er det benyttet kun Microsoft teknologier og verktøy derfor er mange av referansene som blir brukt hentet fra Microsofts nettsider. De beste kildene blir nevnt først.

Windows Media websidene inneholder mye informasjon om de ulike teknologiene som benyttes for en Video on Demand løsninger med Digital Rights Management[6]. Nettstedet inneholder også mange gode tekniske artikler som inneholder nyttig informasjon om VoD og DRM teknologien fra Microsoft.

Microsoft Developer Network er et annet nettsted hvor en finne mye informasjon om DRM teknologien til Microsoft[7]. På MSDN finner en også API referanse

materiale som kan brukes i utvikling av VoD systemer med DRM eller plugins til streamingserveren.

Dokumentasjonen til Windows Media Rights Mangement SDK[8] inneholder mye innformasjon om DRM løsningen fra Microsoft. Den gir god oversikt over hvordan DRM COM objektet fungerer.

Internet Engineering Task Force er et nettsted som inneholder RFC'er og annen nyttig informasjon som kan være nyttige når en lager løsninger som er rettet mot Internett[9]. Blant annet finner en RFC'er om de forskjellige protokollene som benyttes i VoD systemet.

1.4 Rapportstruktur

Målgruppen for rapporten er studenter og software utviklere med basiskunnskaper innenfor .NET utvikling og streamingteknologi. De interesserer seg for distribusjon av digitaltinnhold over Internett og ønsker å vite mer om hvordan en beskytter innholdet med Digital Rights Management.

Kapittel 2 gir en innføring i teorien og teknologien bak DRM fra Microsoft. Det gis en grundig innføring i de viktigste komponentene og prosessene for et VoD system med DRM.

Kapittel 3 og 4 presenterer henholdsvis arkitektur og design for Video on Demand systemer som benytter DRM for å beskytte innholdet. Arkitekturen legger til rette for en automatisert publiseringsprosess. Kapittel 5 tar for seg litt om implementasjon av prototypen.

Kapittel 6 presenterer fordeler og ulemper ved ulike distribusjonsalternativer for innhold. Ulike streaming alternativer og nedlastingsalternativer blir beskrevet nærmere.

Til slutt i kapitlene 7, 8 og 9 blir resultatene presentert og diskutert, før det blir presentert en konklusjon om hvordan DRM arkitekturen fungerer og hvilke distribusjonsalternativ som anbefales for distribusjon av innhold.

2 Teori for Digital Rights Management i Video on Demand systemer

2.1 Bakgrunn

Kapitlet inneholder en introduksjon til teknologiene som ble brukt i utviklingen av Video on Demand systemet.

VoD konseptet går ut på å gi brukere det innholdet de ønsker når de måtte ønske det. For at dette skal være mulig trenger en et sett med teknologier både på klient og på serversiden av systemet. Arbeidet i denne oppgaven baserer seg på den nye Windows Media 9 Series teknologien som ble lansert av Microsoft i januar 2003. Valget falt på denne teknologien fordi iCanal per i dag har en eldre streaming plattform som er levert av Microsoft og jeg ønsket å se på hvilke nye muligheter den nye streaming plattformen fra Microsoft gir. Den er Microsoft største satsing på streamingteknologi noensinne. Rundt 500 millioner dollar er brukt over de 3 siste årene for å utvikle ny kodek, player, enkoder, streamingserver og DRM delen av systemet.

2.2 Innføring i teknologier for Video on Demand

2.2.1 Kodek

Den nye kodeken som kommer med Windows Media 9 Series gir en forbedring på 15 til 50 prosent i forhold til den eldre Windows Media 8 kodeken. Den største gevinsten oppnås ved høye bitrater. Microsoft skriver på sine nettsider at WM 9 Series gir best videokvalitet for alle bitrater uansett hvilke format en sammenligner mot. For eksempel vil en WM 9 fil typisk være halvparten så stor som en MPEG-4 fil med samme videokvalitet. WM 9 har også støtte for 5.1 surround lyd og høydefinisjonsvideo per i dag opp til 720p.

2.2.2 Player

Playeren som brukes er Windows Media Player 9 Series, men eldre versjoner ned til versjon 6.4 vil også fungere. Dersom brukeren ikke har den nyeste Windows Media Playeren trenger en å laste ned de nye kodekene for at den gamle playeren skal fungere. For å få best mulig utbytte av de nye funksjonene som kommer med 9 Series versjonen må en ha Windows XP og et lydkort som støtter 5.1 surround lyd. En kan da nyte filmer med fullverdig 5.1 surround lyd.

Fordelen med Media Playeren er at den er enormt utbredt siden den kommer med alle Windows versjoner som standard. Ved å basere seg på denne playeren vil en

kunne nå store kundemasser uten at kunder trenger å installere nye playere. Det finnes også andre leverandører av playere som støtter Microsofts DRM teknologi så en kan også benytte playere fra disse leverandørene dersom det er et ønske.

2.2.3 Streamingserver

Selve hjertet i VoD systemet er streamingserveren, det er denne som sender innholdet ut til de ulike klientene. Serveren kommer sammen med alle utgavene av Windows .NET Server 2003. Media serveren har gjennomgått en total redesigning fra forrige versjon for å gi bedre pålitelighet, ytelse og skalerbarhet. Mediaserveren baseres også på en plugin arkitektur noe som gir utviklere bort imot frie tøyler til å tilpasse serveren etter deres behov. I kapittel 4 og 5 er det beskrevet nærmere hvordan en lager en slik plugin som forbedrer den standard logging funksjonen som finnes i serveren.

2.2.4 Microsoft DRM

Når en bruker laster ned en kryptert media fil fra for eksempel en webserver eller streamer fra en streaming server, må han skaffe seg en lisens som inneholder en nøkkel som kan låse opp filen før innholdet kan spilles av. Innholdsleverandører kan enkelt beskytte innholdet sitt ved å bruke Windows Media Rights Manager SDK[8].

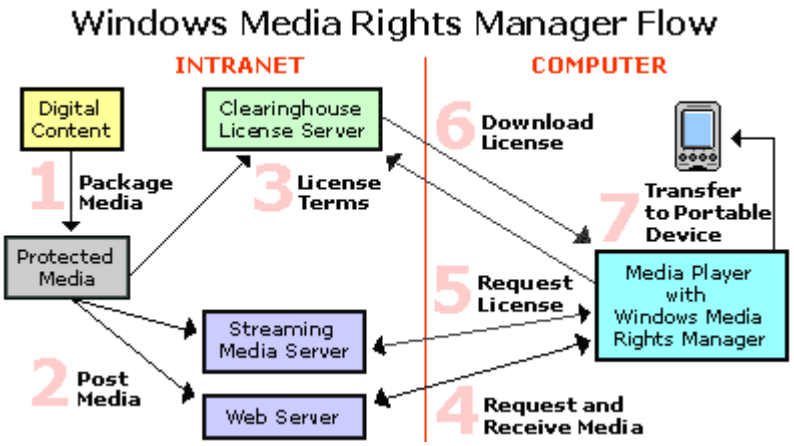
Tidligere versjoner av Microsofts DRM produkt var et ferdig serverprodukt, men de hadde dårlige erfaringene med dette. Brukere strippet bort blant annet GUI og annen funksjonalitet for å tilpasse DRM komponentene sitt system. Derfor har DRM produktet fra Microsoft siden versjon 7 ikke vært et ferdig serverprodukt fordi de regner med at de fleste trenger å gjøre mye spesialtilpassing av DRM løsningen for å integrere med eksisterende systemer. Det kan være eksisterende content management systemer eller eksisterende webapplikasjoner.

Microsoft leverer i dag sitt DRM produkt som et COM objekt en kan programmere mot. Dette gir programmere god fleksibilitet både til å integrere mot eksisterende systemer eller utvikle systemer fra grunnen av. COM objektet tilbyr en rekke interface'er som tilbyr metoder for pakking av innhold, lisensgenerering osv.

Denne oppgaven baserer seg på Windows Media Rights Management 7 SDK'en. Valget ble gjort på bakgrunn av at Windows Media Rights Management 9 Series SDK'en ikke var lansert når arbeidet startet. Forskjellene mellom versjonene er ikke så store, 9 Series versjonen har nye funksjoner knyttet til Live DRM. Live DRM er at en kan pakke innhold direkte i enkoderen mens en koder innholdet. Dette kan være både livebroadcast innhold og on Demand innhold. Fordelen med dette er at en ikke trenger å pakke innholdet etter at det er kodet siden dette gjøres i

samme prosess. Den siste versjonen av SDK'en har også støtte for noen nye rettigheter en kan sette i lisensene.

Figuren 2 viser en overordnet oversikt over hvilke komponenter og funksjonalitet en må ha i et VoD system som benytter DRM teknologi fra Microsoft. Som en ser av figuren trengs pakkeserver, lisensserver, streamingserver, webserver og en playerapplikasjon. De fire serverkomponentene utgjør kjernen av VoD og DRM systemet, mens playerapplikasjonen er klientsoftware som brukes til å spille av innholdet. I tillegg kommer en database for lagring av lisensdata. Pakkeserveren pakker(krypterer) filene slik at de blir beskyttet med DRM. Lisensserverer deler ut lisenser til innhold som brukeren enten streamer fra en streamingserverer eller laster ned fra en webserver.



Figur 2: Arkitekturen til DRM systemet fra Microsoft.

Figuren 2 viser i tillegg til hovedkomponentene for DRM systemet en oversikt over de viktigste prosessene i systemet. Under følger en forklaring til de ulike prosessene et DRM system som baserer seg på teknologi fra Microsoft må inneholde. Prosessene er nummeret fra 1 til 7 i figur 2.

1. Pakking av innhold
   Innenfor DRM så kalles krypteringsprosessen av filene for pakking av mediafilene. En benytter Windows Media DRM SDK'en til å pakke Windows Media filer (*.wmv eller *.wma) uten DRM beskyttelse, til Windows Media filer med DRM beskyttelse. Media filene blir i denne prosessen låst med en nøkkel. Nøkkelen lagres i en kryptert lisens som da leveres separat fra resten av filen. Pakkingen av filen legger til en header som inneholder en del informasjon om mediafilen som for eksempel url'en til der hvor brukeren kan få tak i en lisens til filen. Den som utfører pakkingen kan også legge til annen informasjon i filen som for eksempel

hvem som har pakket filen, copyright informasjon eller for annen egendefinert informasjon.

2. Distribusjon av mediafiler
   Det finnes flere ulike alternativer for å distribuere innholdet til brukerne. De DRM beskyttete filene kan distribueres ved å legge dem ut på webservere for nedlasting, legge dem på streaming servere for streaming, distribueres på CD'er eller sendes i e-post til brukere. For en Video on Demand løsning vil først og fremst streaming brukes på grunn av at det gir brukeren den beste "on Demand" løsningen men nedlastingsalternativer kan også brukes.
3. Lisensservere
   Innholdsleverandørene må velge en aktør som kan distribuere lisenser til brukerne. Dette vil typisk ligge hos en aksessleverandør. Ved etablering av en lisensserver er det viktig å spesifisere hvilke rettigheter som skal gis i de forskjellige lisensalternativene. Oppgaven til lisensleverandøren er å autentisere brukerne som forsøker å få tak i en lisens. Mediafilene og lisensene lagres og distribueres separat noe som gjør systemet enkelt å vedlikeholde. En kan for eksempel dele ut forskjellige lisenser med ulike rettigheter til en og samme fil.
4. Innholdsnedlasting
   Brukeren sender en forespørsel etter innholdet. Innholdet kan enten streames fra innholdsleverandør eller lastes ned fra en webserver.
5. Lisens anskaffelse
   For å spille av den DRM beskyttete filen må som nevnt tidligere brukeren skaffe seg en lisens som inneholder nøkkelen til å låse opp filen. Det finnes ulike metoder for å levere lisenser til brukerne:
   a. Predelivery – Lisensen leveres til brukeren før han prøver å spille av filen. Lisensen vil typisk leveres når brukeren kjøper og betaler produktet. Når brukeren ønsker å spille av filen finnes den allerede på maskinen og brukeren vil ikke merke noe til lisensieringsprosessen.
   b. Silent lisenslevering – lisensen lastes ned automatisk uten av bruker må registrere seg eller betale for innholdet.
   c. Nonsilent lisens levering – Når bruken forsøker å spille av mediafilen må han først gjennomføre en form for registrering eller betaling før han mottar lisensen.
6. Lisens lastes ned
   Brukeren kan da spille av innholdet. For å kunne benytte DRM beskyttet innhold må brukeren ha en player som støtter Windows Media Rights Manager. Dersom brukeren har det er maskinen i stand til å bruke mediafilen i henhold til rettighetene som er spesifisert i lisensen han har mottatt. Rettigheter som kan settes i lisensen for å begrense bruken kan for

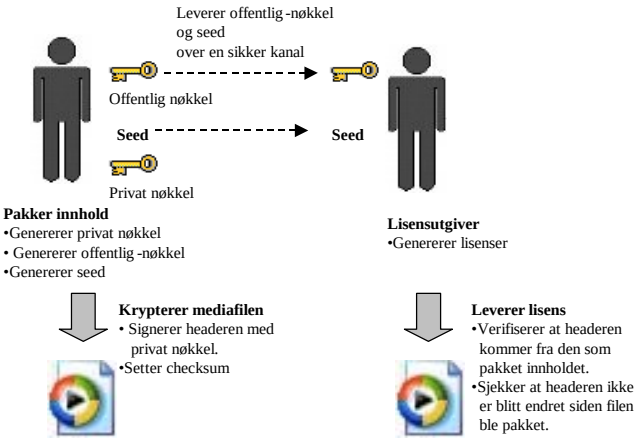
eksempel være startdato, sluttdato eller antall avspillinger. Se avsnitt 2.3 for mer informasjon om rettigheter som kan settes i lisensene. Det er viktig å bemerke at lisenser er individuelle, det er ikke mulig og kopiere en lisens og sende den til andre. En lisens er kun gyldig på maskinen hvor den først ble installert. Dersom en sender en DRM beskyttet fil til andre må disse selv skaffe seg sin egen lisens.

7. Filene kan overføres til bærbare enheter dersom det er ønskelig og tillatt i henhold til rettighetene spesifisert i lisensen. Bærbare enheter kan være alt fra PDA'er til musikk "jukebokser".

2.2.5 Nøkkel og seed generering

DRM teknologien fra Microsoft benytter ulike nøkler og seed for å beskytte innholdet. Nøklene som benyttes er asymmetriske det vil si at de er matematisk relaterte, men to helt ulike nøkler. Nøklene blir ofte kalt hhv offentlig og privat nøkkel. De benyttes for å lage digitale signaturer, signaturene lages ved hjelp av RSA algoritmen. Nøklene benyttes sammen med seed'et til å pakke innhold og i genereringen av lisenser. Dersom en skal pakke en fil med DRM fra Microsoft må en ha en privat nøkkel og et seed som brukes i pakkingen. Seed'et brukes i selve krypteringen, mens den private nøkkelen brukes til å signere header'en i filene som pakkes med en digitalsignatur. Et seed er en "delt hemmelighet" som både organisasjonen som pakker og organisasjonen som leverer lisenser til innhold må ha kjennskap til for at systemet skal fungere. Det er en tilfeldig generert tekststreng på minst 5 bytes. Strengen brukes av DRM krypteringsalgoritmen som random generator. Fordelen med bruk av nøkler er at en kan verifisere at headeren virkelig er generert av den som pakket filen. Det gjøres ved bruk av RSA digitale signaturer. Når en mottar en lisensforespørsel fra en klient kan en sjekke at headeren er generert av den som pakket filen. Det gjøres ved å bruke den offentlige nøkkelen til å verifisere at headeren er generert av riktig organisasjon før en leverer en lisens til brukeren. Dette blir gjort for å øke sikkerheten i systemet, en leverer ikke ut lisenser uten at en er sikker på hvem som pakket filen. En unngår slik noen former for hacking. For at dette skal være mulig må organisasjonen som skal levere lisensene ha kjennskap til den offentlige nøkkelen til den som pakket filen. Distribusjon av denne informasjonen bør skje over sikre kanaler. Det kreves at headeren signeres når en pakker innholdet, men det er ikke et krav at en autentiserer opphavet av headeren før en leverer ut lisenser. Men jeg vil anbefale at en autentiserer headeren før en leverer lisenser til brukere. Det minsker mulighetene for å hacke løsningen. Det regnes også ut en checksum over keyen som genereres under pakkingen. Det blir gjort for at en skal forsikre seg om at headeren som inneholder key'en ikke er blitt endret av andre.

DRM COM objektet som levers av Microsoft har metoder for å generere nøkkel par og seed. Siden den offentlige og den private nøkkelen er avhengige av hverandre må de genereres av samme part. Det er den som pakker innholdet som vil stå for å generere nøklene. Seed'et genereres også av den som pakker innholdet og sendes lisensutgiveren sammen med den offentlige nøkkelen.



Figur 3:Oversikt over hvordan seed og nøkkel paret privat og offentlig nøkkel brukes.

Figuren over viser at den som pakker innholdet genererer nøklene og seed'et som brukes i pakkingen av innholdet. Offentlig nøkkel og seed leveres lisensutgiver over en sikkerkanal slik at lisensutgiveren er i stand til å levere ut lisenser til innholdet. Det er viktig å merke seg at det ikke nødvendigvis må være to forskjellige organisasjoner som står for pakking og levering av lisenser. Pakkingen og lisensieringen kan også gjøres av en og samme organisasjon.

2.2.6 Pakking av mediefiler

Som nevnt tidligere så krypteres media filene under pakke prosessen. Krypteringsalgoritmen som brukes er meget effektiv, algoritmen er i stand til å kryptere 10 MB per sekund. Krypteringen fører heller ikke til at filen blir noe nevneverdig større i volum, det legges til en header på filen men den er så liten at den ikke har stor innvirkningen på filstørrelsen. For eksempel var en fil 242 KB før pakking og etter pakking ble den 243 KB, størrelsen økte med kun 1 KB. Det samme var tilfelle med filer som var på flere hundre MB.

Det finnes to forskjellige alternativer for pakking av innhold, det er batch pakking og normalpakking av innholdet. Pakkeprosessen i seg selv er ikke forskjellig men måten en benytter nøkkel par, keyid og seed. For normal pakking benytter en forskjellige nøkkel par, keyid og seed for hver enkelt fil en pakker. Det genereres nye nøkkel par, keyid og seed for hver fil som pakkes. For batch pakking benytter

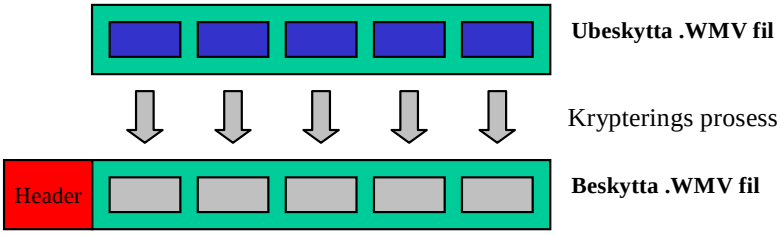
en de samme verdiene for nøkkel par, keyid og seed for flere filer. En trenger da kun en lisens for alle filene som ble pakket med de samme verdiene. Hvordan en pakker innholdet bestemmes av hvilke type innholdet det er. Normal pakking av innhold brukes for Pay-Per-View innhold, mens batch pakking benyttes for innholdsabonnementer.

Headeren som legges til inneholder en del viktig informasjon som brukes under lisensieringen av filen. Noen av feltene er obligatoriske andre er valgfrie. Listen under tar for seg de viktigste feltene, fullstendig liste over attributter og metoder finner en i WMRM SDK:

- Key ID attributtet er en unik id som brukes når en skal generere en lisens som inneholder nøkkelen til å låse opp filen. Attributtet er obligatorisk.
- Content ID attributtet inneholder en id som unikt definerer det krypterte innholdet. Attributtet er ikke obligatorisk, men det er anbefalt at en har med denne. For eksempel så kan en bruke attributtet når en setter rettigheter i lisensen. En kan ha en database hvor en registrerer rettigheter som skal settes på de forskjellige Content ID'ene. En kan også bruke samme content ID på alle filer som tilhører samme innholdsleverandør. Det er da enkelt å skille innhold fra forskjellige innholdsleverandører for eksempel hvis en opererer med forskjellige business regler for forskjellige innholdsleverandører.
- En kan også legge til så mange egendefinerte attributter en måtte ønske i headeren. Microsoft anbefaler dette fordi dette kan øke innholdsverdien til filen. En kan for eksempel legge til url til den offisielle hjemmesiden til filmen eller artisten osv.
- LicenseAcqURL attributtet inneholder url'en hvor en kan få tak i en lisens til filen. Dersom en ikke benytter predelivery av lisenser eller at brukeren av en eller annen grunn har mistet sin lisens til filen er det viktig at en kan finne ut hvor en kan få tak i en lisens. Dersom en bruker mangler lisens til en fil har Windows Media Playeren funksjonalitet for å benytte dette attributtet for å få tak i en lisens. Lisensvarehuset kan da for eksempel implementere betalingsløsninger hvor brukeren betaler for innholdet før lisensen lastes ned.

Figur 4 viser hvordan media filen deles i biter når den krypteres og hver bit blir kryptert hver for seg. Segmentene av filen er ikke avhengige av hverandre, noe som er en stor fordel når det gjelder streaming scenarioer. Dersom en på grunn av pakketap mister segmenter under overførselen av filen kan en om nødvendig droppe segmentene uten at dette forstyrrer dekrypteringen. Dersom pakkene hadde vært avhengige av hverandre måtte en mottatt alle segmentene og retransmisjon av segmenter som mistes hadde tatt mye tid og i verste fall ført til at bufferet hos

klienten hadde gått tomt. Det hadde også stridd mot streaming prinsippene hvor pakketap kan aksepteres, dersom pakker går tapt så vil playeren droppe å rendre framene som gikk tapt på grunn av pakketapet. Streamen stoppes ikke selv om ikke alle pakkene mottas i tide.



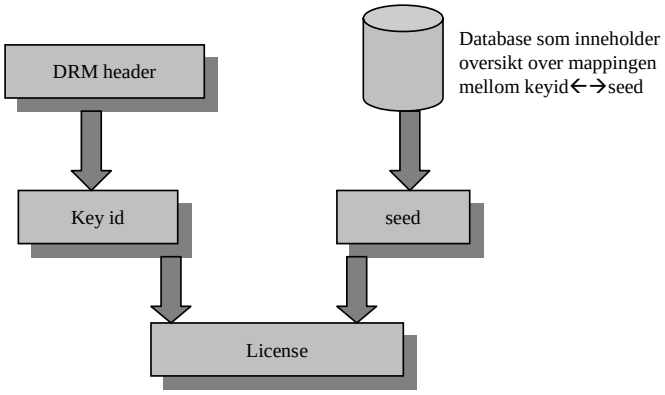
Figur 4: Oversikt over krypteringen en mediafil.

DRM beskytta materiale blir lagret i en kryptert form på harddisken eller på andre lagringsmedier. Krypteringsalgoritmen koder dataene i filene, og på denne måten gjør den dataene ubrukelige inntil de blir dekryptert. Under avspilling av innholdet forblir innholdet kodet selv når det lastes inn i minnet. Audio dataene forblir kodet helt til slutten av ”data stien” hvor DRMK system driveren dekrypterer innholdet og sender det direkte til lydkort driveren slik at det kan spilles av. Ved å gjøre stien hvor dataene er ukrypterte kortest mulig gjør DRMK innholdet mindre sårbart for uautorisert kopiering. I Windows 98 og Windows 2000 finnes det et sikkerhetshull som gir brukere muligheten til å få tak i dekryptert materiale. For Windows Me og XP og nyere versjoner er dette sikkerhetshullet tettet. I Windows Me, Windows XP og nyere versjoner forblir innholdet kryptert mens det traverserer audio-data stien inntil det når et beskytta område i kjernen av operativsystemet. I kjernen dekrypteres innholdet og leveres til en betrodd audio driver.

En forutsetning som er tatt i arkitektur, design og implemtasjons kapitlene er at innholdsleverandører leverer sitt materiale ubeskyttet til aksessleverandøren. Dersom innholdsleverandørene hadde levert sitt innhold beskytta hadde publiseringsprosessen vært en del enklere. Da ville innholdsleverandøren selv stått for pakkingen av innholdet og en utveksling av offentlig nøkkel og seed måtte blitt gjort over en sikker kanal.

2.2.7 Lisensgenerering

De sentrale elementene i lisensgenereringen er key id’en som hentes fra headeren til mediafilen og seed’et som ble brukt til å pakke innholdsfilen. Ved hjelp av key id’en og seed’et er en i stand til å generere lisenser som kan ”låse” opp den DRM beskytta filen.



Figur 5:Overordnet oversikt over elementene som trengs for å generere DRM lisenser

Figuren over viser en overordnet oversikt over hvordan en generer lisenser. Ved å hente ut key id'en fra headeren og seed'et som ble brukt under pakkingen av filen fra databasen kan en generere lisenser til filen. Prosessen med å generere lisenser høres enkel ut men er rimelig avansert. Lisensprosessen involverer en rekker av objektene som kommer med Microsofts DRM løsning.

Som en ser av figur 5 så er det veldig viktig at seed'et behandles sikkert. Dersom noen får tak i seed'et så kan en generere egne lisenser for innholdet. Seed'et bør lagres i en database og kun være tilgjengelig til de som må ha tilgang til det. Dersom en har et system hvor pakking og lisenslevering er to ulike organisasjoner må en etablere en sikker kanal for å overføre informasjonen om seed'et. For eksempel kan en benytte HTTPS eller andre sikre overføringsmetoder.

Det er viktig å sjekke informasjonen klienten sender i lisensforespørselen før en genererer lisenser. Klienten sender blant annet informasjon om hvilke klient som sender forespørselen, versjonsnummer, om playeren er individualisert eller ikke. Informasjonen bør sjekkes slik at en ikke sender lisenser til for eksempel klienter som er blitt hacket. Dersom det er krav om at klienten skal være individualisert må også dette sjekkes. Sjekkene gjøres for å gjøre DRM systemet så sikkert som mulig.

Lisenser slettes når de ikke er gyldige lenger enten ved at tids- eller antall avspillinger begrensingene er overskredet. For eksempel dersom en lisens gir begrensinger som 15 avspillinger og to overførsler til bærbart utstyr må begge disse begrensingene være overskredet før lisensen slettes. Dersom kun en av begrensingene er nådd så vil lisensen lagres inntil den andre begrensingen er

overskredet. Dersom en har tidsbegrensinger i lisensen vil lisensen slettes dersom enten en antall begrensing eller tidsbegrensingene overskrides.

Som nevnt tidligere kan innholdet pakkes på to forsjellige måter. Vanlig pakking for Pay-Per-View innhold og batchpakking for innholdsabonnementer. Fordelen med å skille de to pakkingsalternativene trer frem når en skal levere lisenser til innholdet. For normalt pakket innhold trengs en lisens per mediafil, mens for batch pakket innhold kan en lisens benyttes for flere filer. Siden batch pakket innhold benytter de samme verdiene for nøkkel par, keyid og seed trenger en kun en lisens for filene som er pakket med de samme verdiene. Alt innhold innen et abonnement pakkes med samme nøkkel par, keyid og seed. En bruker trenger derfor kun en lisens for alle filene i et abonnementet.

2.3 Lisensrettigheter

Rettighetene som settes i lisensene spesifiserer hvordan en bruker kan benytte en Windows Media fil. For eksempel hvor lenge eller hvor mange ganger en bruker kan spille av innholdet som dekkes av lisensen. Rettigheter kan også bestemme hvordan brukere kan behandle filen dvs. om den kan kopieres til CD, spilles på PC eller brukes på bærbart utstyr osv.

Det er spesielt en rettighet som gjelder bruken av filen en skal være klar over hvordan fungerer. AllowBackupRestore rettigheten kan misbrukes dersom den ikke benyttes riktig. Dersom en har satt en antallsbegrensning i lisensen og tillater backup av lisensen må en være klar over at bruk av AllowBackupRestore rettigheten kan føre til svindel med rettigheter. Når en bruker har en lisens som tillater fem avspillinger av en media fil kan brukeren ta en backup av lisensen når han har fem avspillinger igjen i lisensen. Brukeren kan da spille av filen fire ganger og deretter gjenopprette backup versjonen av lisensen, brukeren vil da ha fem avspillinger igjen på filen istedenfor en. En må være klar over hvordan rettighetene fungerer sammen slik at kombinasjonen av rettigheter som settes i en lisens gjør at brukere ikke kan misbruke lisensene.

2.4 Windows Service

Publiseringsalternativene som beskrives i denne oppgaven benytter Windows servicer, dette avsnittet er ment å gi en kort beskrivelse av slike servicer. Windows plattformer har støtte for å lage egne servicer som kjøres under Windows. Det er applikasjoner som kjøres over lengre tid det vil si at de trigges ved gitte tidsintervaller. Services kjøres i egne Windows sesjoner og kan blir startet

automatisk nå maskinen starter eller startet manuelt. Servicer kan også settes i pause modus, stoppes eller restartes. Servicer er ideelle for bruk på servere hvor en trenger applikasjoner som kjøres over lang tid.

Ved bruk av .NET rammeverket kan en enkelt utvikle egne servicer. Det finnes en base klasse, System.ServiceProcess.ServiceBase, som har metodene som trengs for at programmet skal fungere som en Windows service. Det en trenger i tillegg er en installer som installerer og oppdaterer registeret på maskinen hvor servicen skal kjøres slik at servicen kan kjøres på maskinen.

Den enkleste implementasjonen for å automatisere publiseringen av innholdet er å benytte Windows servicer til å gjøre pakkingen og distribusjonen av innholdet til streaming og/eller webservere. En benytter en Windows service som skrives i .NET (programmeringsspråk er valgfritt: C++, C# eller VB.NET) Innholdsleverandører kan benytte FTP programvare for å laste opp innhold til aksessleverandøren, etter hvert som innholdet blir lastet opp vil det bli pakket, lisensdata lagres i databasen og innholdet distribueres til de ulike streamingserverne ved hjelp av Windows servicer

2.5 Lisenslevering

2.5.1 Predelivery

Det er to av de tre leveringsmetodene for lisenser som vil bli brukt i denne oppgaven. Det er predelivery av lisenser og Nonsilent delivery. Metoden som egner seg best er predelivery, ved bruk av denne vil brukeren ikke merke noe til lisensleveringen. Paktisk så leveres lisensen til brukeren i det øyeblikket at et kjøp er bekreftet. Lisensen lastes da ned på brukerens PC og når streamingen eller avspillingen av innholdet starter er lisensen allerede på plass. Fra brukerens ståsted så er han sannsynligvis ikke interessert å sette seg inn i hvordan lisenser til innhold fungerer i praksis, han er ute etter å benytte innholdet som han har betalt for. Dersom en skal designe en løsning som har brukervennlighet i fokus bør en benytte denne formen for levering av lisenser.

2.5.2 Nonsilent lisenslevering

Som et supplement til predelivery av lisenser må en ha nonsilent levering av lisenser. Dersom en bruker av en eller annen grunn har mistet sin lisens ved at den er blitt slettet, harddisk krasj eller lignende må en ha en metode for å levere lisenser til disse brukerne. Predelivery leverer lisensene i kjøpsdialogen i det kjøpet er bekreftet, men dersom brukeren har betalt for et produkt så ønsker brukeren ikke å kjøpe produktet på nytt kun fordi lisensen er slettet. Dette løses ved å tilby en

nonsilent levering av lisensene til brukeren. Når brukeren prøver å starte en fil som han mangler lisens til vil Mediaplayeren sjekke headeren og finne url adressen hvor brukeren kan få tak i en ny lisens til filen. Lisensleverandøren må da implementere en løsning hvor brukeren kan logge inn og få tak i en ny lisens til filen. Se vedlegg D. Denne løsningen for lisenslevering kan også brukes dersom brukeren for eksempel har mottatt filen fra andre brukere. Aksessleverandøren implementerer da en løsning for betaling i urlen som er lagret i headeren til filen. Brukeren får da presentert en betalingsdialog før lisensen lastes ned.

2.6 Protokoller

Det benyttes en del forskjellige protokoller for de ulike publiseringsalternativene. Arkitekturen som oppgaven beskriver benytter følgende protokoller for å distribuere innhold til sluttbrukerne:

- HTTP/1.0 Hypertext Transfer Protocol versjon 1.0 [10]
- HTTP/1.1 Hypertext Transfer Protocol versjon 1.1 [11]
- RTSP Real Time Streaming Protocol. [12]
- MMS Microsoft Media Server protocol. [13]

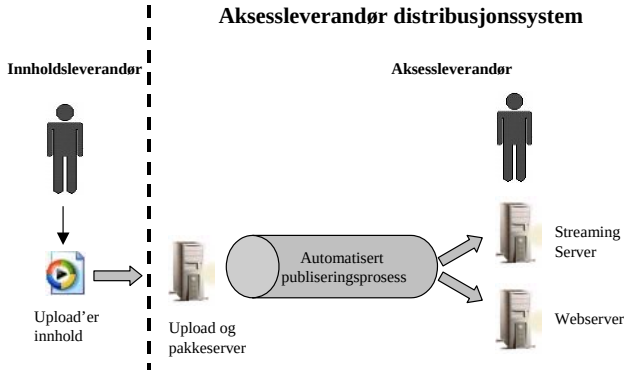
FTP, File Transfer Protocol, protokollen brukes til å distribuere innhold som er ferdig pakket fra pakkeserveren til streamingserverne og/eller webservere[14].

3 Prototyp arkitektur

3.1 Bakgrunn

Som nevnt i innledningen består DRM systemet fra Microsoft av et COM objekt som inneholder en rekke interface'er en kan benytte for å pakke og levere lisenser til innhold. For at en organisasjon skal kunne benytte DRM i sine løsinger trengs en innpakking rundt COM objektet. DRM produktet fra Microsoft er ikke et produkt som er klart til bruk. En trenger forskjellige applikasjoner for å pakke og levere lisenser til innhold. Applikasjonene benytter metodene i COM objektet for å utføre pakking og lisensiering. Organisasjoner som ønsker å benytte DRM må selv lage sine egne tilpassa applikasjoner for å utføre de ulike oppgavene i et DRM system.

Vi vil i de neste kapitlene se på hvordan en kan sette sammen et system for pakking og lisensdistribusjon. Det er lagt vekt på at det skal være mye automatikk i systemet. Innholdsleverandører laster opp innhold til aksessleverandører og innholdet blir pakket og distribuert til streamingserver og/eller webservere. Publiseringsprosessen vil være automatisert det vil si at pakking, lagring av lisensdata og distribusjonen av innholdet til streamingserver og/eller webservere er automatisert.



Figur 6: Skisse over automatisert pakkeprosess.

Figuren over illustrerer at innholdsleverandører laster opp innhold til aksessleverandører. Når innholdet er lastet opp blir det behandlet av en automatisert pakkeprosess hvor innholdet blir pakket, lisensdata lagret og innholdet distribueres til streamingservere og/eller webservere.

Oppsettene av Video on Demand systemer med DRM kan variere mye. De ulike oppsettene vil oppfylle ulike krav til oppetid for systemet, drifts og vedlikeholds fordeler og ulike kostnadsrammer. Hovedkomponentene i systemet er lisensserver, streamingserver, pakkeserver og webserver. Hvordan en setter opp systemet er

opptil hver enkelt organisasjon. Mest hensiktsmessig vil det nok være å kjøre de ulike komponentene på forskjellige servere. Det vil gi fordeler når det gjelder drift og vedlikehold av systemet. Dersom en for eksempel må vedlikeholde lisensserveren kan denne tas ned uten at en trenger å ta ned andre servere. Brukere som allerede har lastet ned lisens vil ikke bli berørt av en slik vedlikeholdsjobb og kan benytte innholdet som ligger på streaming eller webserveren. Brukere som mangler lisenser får ikke tak i det før lisensserveren er oppe igjen. Arkitekturen gir stor fleksibilitet til systemet når det gjelder drift og vedlikehold. Dersom krav til oppetid er enda større kan en kjøre clustering på de ulike komponentene. En rimeligere konfigurasjon er å kjøre flere av komponentene på en og samme maskin. I prinsippet kan en kjøre alle komponentene på samme maskin, en sparer da kostnader på at en trenger kun en maskin istedenfor fire som i forrige eksempel. Oppsettet er dårlig sett fra et drifts og vedlikeholds synspunkt. Dersom en skal vedlikeholde eller oppgradere en av komponentene kan dette ha innvirking på alle de andre komponentene hvis en f.eks. trenger en omstart av maskinen. Hvordan en setter opp systemet bestemmes av hver enkelt organisasjon ut ifra krav til oppetid, drift og kostnadsrammer.

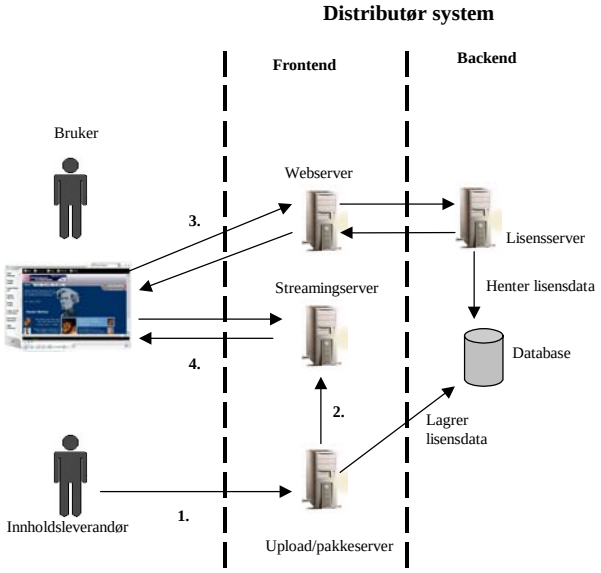
I gjennomføring av denne oppgaven ble det brukt to pc'er for å operere som henholdsvis streamingserver og webserver/lisensserver/ uploadserver/pakkeserver.

Tabell 1: Oppsettet av test PC'er brukt i gjennomføringen av oppgaven.

PC spesifikasjon		**Oppgave**
Maskin	Bærbar Compaq Armada 110	Streamingserver
CPU	Celeron 700 MHz	
Ram	256 MB	
Operativsystem	Windows .NET Server 2003	
Streamingserver	Windows Media Services 9 Series	
Maskin	Bærbar Compaq Armada M700	Webserver/lisensserver/ uploadserver/ pakkeserver
CPU	P3 850MHz	
Ram	512 MB	
Operativsystem	Windows 2000	
Web server	Internet Information Server 6.0	

Komponentene som er utviklet for prototyp DRM systemet er selvstendige komponenter i den forstand at de kan kjøre på forskjellige maskiner. Komponentene kommuniserer sammen ved hjelp av blant annet HTTP, FTP, SOAP og ulike streaming protokoller som RTSP og MMS. Figuren under viser en grovskisse over hvordan systemet er satt opp. Http brukes til nedlasting av innhold fra en webserver, mens RTSP og MMS benyttes dersom innholdet streames fra en

streamingserver. SOAP brukes blant annet for kommunikasjonen mellom lisensserverens webservice og webserveren som leverer lisensene til brukerene. FTP benyttes for å laste opp innhold fra innholdsleverandører til aksessleverandørenes uploadserver. Det benyttes også FTP for å distribuere ferdig pakket innhold fra pakkeserveren til streamingservere og/eller webservere.



Figur 7: Grovskisse over arkitekturen av prototypsystemet

Kort forklaring til de ulike punktene i figuren:

1. Innholdsleverandør laster opp innhold til aksessleverandøren. Vi skal se på to ulike alternativer for uploading av innhold. Alternativ en er å uploade innhold ved hjelp av vanlig FTP programvare og alternativ to er å lage en tilpasset uploadingsklient.
2. Innholdet pakkes, lisensdata lagres i databasen og innholdet distribueres til streamingservere og/eller webservere automatisk med komponentene laget for prototypen.
3. En bruker ønsker tilgang til innholdet, han må først gjennom en betalingsdialog for å betale for innholdet. Etter at han har betalt for innholdet kan brukeren motta en lisens som gir han tilgang til å benytte innholdet.
4. Brukeren har en lisens til innholdet installert på maskinen og kan da starte streaming eller progressiv nedlasting av innholdet. (Dersom en baserer seg på normal nedlasting av innholdet kan brukeren først laste ned innholdet før han blir bedt om å skaffe seg en lisens dersom predelivery ikke brukes.)

For å implementere arkitekturen skissert i figur 7 er det utviklet følgende komponenter:

- DRM rammeverk – rammeverk for DRM støtte.
- Pakkeservice – Windows service for å pakke innhold.
- Distribusjonsservice – Windows service for å distribuere innhold fra pakkeserver til streamingservere og/eller webservere
- Lisensserver webservice – Webservice som genererer de ulike lisenstypene.
- Webapplikasjon – En enkel demonstrasjons webapplikasjon som viser bruk av de forskjellige lisensieringsalternativene.
- Uploadingsklient – klient som brukes av innholdsleverandører til å sette rettigheter, velge pakkealternativ og distribusjonsalternativ for innhold som lastes opp til aksessleverandør.

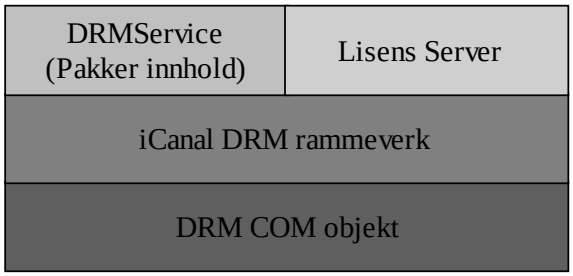
3.2 DRM Rammeverk

Som nevnt tidligere består Microsoft sitt DRM produkt av et COM objektet som tilbyr en rekke interface'er som utviklere står fritt til å benytte. Det er fullt mulig å benytte COM objektet sine interface'er direkte i pakke eller lisenskomponentene, men interfacene er laget for generell bruk. Jeg valgte å lage et rammeverk som bygger på COM objektet. I tillegg til basisfunksjonaliteten til COM objektet er det lagt til noe ny funksjonalitet i dette rammeverket. All errorhåndtering håndteres av rammeverket noe som gjør implementasjonen av rammeverket enklere. Dersom det oppstår error situasjoner vil rammeverket logge dette. Rammeverket logger alle feil som oppstår og i tillegg vil en del debugging informasjon logges. Loggene skrives til fil. Den største fordelen med rammeverket er factory klassene som forenkler pakke og lisensprosessene. Factory klassene tilbyr metoder for å pakke og generere lisenser til innholdet.

Tanken bak rammeverket var at de ulike komponentene som inngår i DRM arkitekturen skal benytte rammeverket for å gjøre implementeringen av DRM enklere. For å implementere DRM funksjonalitet så trenger en ikke å kjenne til DRM COM objektet og dets interface'er når en bruker iCanal DRM rammeverket. Det kan være en tidkrevende prosess å sette seg inn i hvordan DRM COM objektet fungerer i detalj. Objekter skal brukes i riktig rekkefølge, ha riktige innparametere og attributter satt for at prosessene skal fungere. Ved å legge til et lag i arkitekturmodellen trenger en programmerer kun å benytte funksjoner som finnes i iCanal DRM rammeverket for å pakke og generere lisenser. Klassene PackagingFactory og LicenseFactory i rammeverket tilbyr funksjoner for å utføre pakking og lisenslevering. En trenger da ikke ha full oversikt over hvordan DRM

prosessene fungerer i detalj. For eksempel dersom en ønsker batch pakking av en fil så kan dette gjøres ved å kalle en metode i iCanal DRM rammeverket. Rammeverket vil pakkefilen og lagre de nødvendige lisensdataene. Tilsvarende finnes det metoder i rammeverket for å generere ulike lisenstyper.

Det er valgt en strict layering arkitektur, det vil si at implementasjonen av et lag kun kan baseres på de operasjonene som tilbys av laget som ligger like under i arkitekturmodellen. Valget er gjort for at errorhåndteringen skal utføres av iCanal DRM rammeverket og for å legge til endel annen funksjonalitet som factory klassene.



Figur 8: Prototypen har en lagvis delt arkitektur for DRM funksjonaliteten.

Figuren over viser arkitekturen for DRM komponentene som er benyttet i prototypen.

- Lag 1 består av DRM COM objektet fra Microsoft.
- Lag 2 består av DRM rammeverket som er laget som en del av denne oppgaven.
- Lag 3 består av Windows service'en som pakker innholdet og lisensserver webservice'en. Begge komponentene utviklet i denne oppgaven.

ICanal DRM rammeverket fungerer sammen med DRM COM objektet fra Microsoft som grunnsteinene når resten av DRM systemet settes opp.

3.3 Oppsett av uploadserver

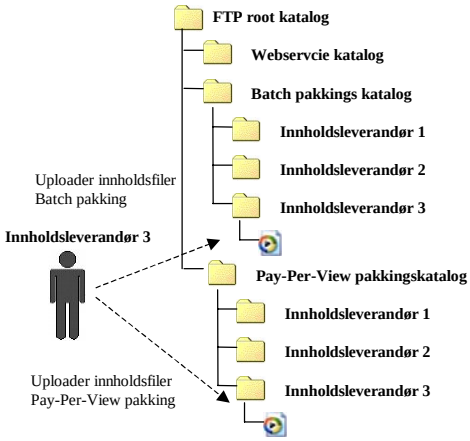
Uploadserveren er mellomleddet mellom innholdsleverandørene og distributøren. Innhold lastes opp på serveren av innholdsleverandører ved hjelp av FTP programvare. Det kan også tenkes at innhold leveres distributøren på annet vis for eksempel på CD eller DVD. Da uploader eller kopierer distributøren selv innholdet til uploadserveren. Uploadserven står også for pakkingen og distribusjon av innholdet til streamingservere og/eller webservere.

Siden det finnes forskjellige pakkingsalternativer og lisensieringsalternativer så har jeg valgt å se på to ulike løsninger for å laste opp innhold. Alternativ en er å laste opp innhold med vanlig FTP programvare og organisere filstrukturen på uploadserveren på en spesiell måte for å skille de ulike pakke og lisensalternativene. Hvordan innholdet pakkes og hvilke rettigheter som settes er avhengig av katalogen det blir lastet opp til siden pakkealternativ og rettigheter er knyttet til kataloger. Alternativ to er å bruke en tilpasset uploading klient hvor brukeren selv definerer hvordan innholdet skal pakkes og hvilke rettigheter som skal settes i lisensene til innholdet.

3.3.1 Uploading av innhold med vanlig FTP programvare.

I alternativ en for uploading av innhold vil innholdsleverandører laste opp innholdet til aksessleverandøren ved hjelp av vanlig FTP programvare. Hvordan innholdet pakkes og hvilke rettigheter som settes i lisensen bestemmes av hvilke katalog innholdet lastes opp til.

Katalogstrukturen på uploadserveren er organisert på en spesiell måte og innholdsleverandører benytter vanlig FTP programvare for å uploade innhold. Innhold som skal pakkes som Pay-Per-View eller batch pakking fra de ulike innholdsleverandørene uploades til forskjellige kataloger på uploadserveren. Hver av disse katalogene har underkataloger som tilhører de forskjellige innholdsleverandørene. Databasen inneholder informasjon om hvilke rettigheter som skal settes for innhold i de forskjellige katalogene. Løsningen er laget slik at dersom en ønsker å legge til flere innholdsleverandører så trenger en kun å legge til en ny underkatalog som skal tilhøre den nye innholdsleverandøren. Og oppdatere rettighetsinformasjonen som skal knyttes til katalogen som ble opprettet i databasen. Når en setter opp FTP serveren så gir en brukere kun skrive rettigheter til katalogen. Det er ingen hensikt i å gi lese rettigheter til katalogene siden innholdet flyttes etter at det er pakket. Uploadområdet brukes som et mellomlager mens opplastingen av filen pågår.



Figur 9: Organiseringen av katalogstrukturen på uploadserveren.

Figuren over viser at innholdsleverandør 3 ønsker å publisere en fil. Dersom innholdsleverandøren ønsker å benytte Pay-Per-View pakking lastes filen opp til katalogen som tilhører innholdsleverandør 3 som ligger i Pay-Per-View katalogen. Ønsker innholdsleverandøren å benytte batch pakking av filen uploades filen til katalogen innholdsleverandøren har tilgang til under batchpakking.

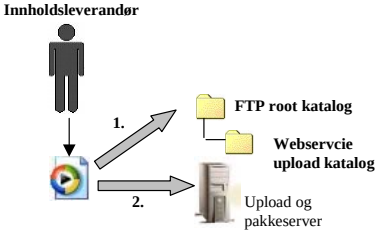
3.3.2 Pakke Prosess

Dersom en skal håndtere store menger med innhold bør en ha en automatisert prosess for å pakke innholdet. En kan tenke seg hvor tidkrevende det hadde vært dersom en hadde basert seg på å pakke alt innholdet manuelt dersom en mottok tusenvis av filer hver eneste uke. En metode for å automatisere denne prosessen er å benytte Windows services.

Prinsippet er enkelt, en lager en service som sjekker om det finnes filer i noen bestemte kataloger på harddisken. Servicen vil da sjekke om det er kommet nye filer i katalogene som brukes til opplasting av innhold fra innholdsleverandører. Servicen sjekker dette ved et spesifisert tidsintervall. Under testing av servicen benyttet jeg et tidsintervall på 10 sekunder, men dersom denne service'en skulle vært brukt i et produksjonsmiljø kunne en operert med mye lenger tidsintervaller, for eksempel fem minutter, for å spare CPU ressurser. Selv om det ikke krever så mye CPU ressurser dersom servicen ikke finner noen filer som skal pakkes. Om publiseringen av innhold tar noen minutter så er ikke dette kritisk. Dersom servicen finner ut at det er lastet opp nye filer fra innholdsleverandører siden sist gang servicen ble kjørt pakkes filene og lisensdata lagres i databasen. Servicen pakker kun filer som ferdig overført. Dersom innholdsleverandører laster opp filer på flere gigabyte så vil ikke filen pakkes før den er ferdig overført. Etter at filen er pakket vil den sendes videre til distribusjonsservicen.

3.3.3 Uploading av innhold med tilpasset programvare.

Alternativ to for å uploade innhold fra innholdsleverandører til aksessleverandører er å benytte en tilpasset klientapplikasjon. Klientapplikasjonen laster opp innholdet ved hjelp av FTP. Den kan i tillegg gi innholdsleverandøren muligheten til å administrere rettighetene på innholdet som lastes opp, bestemme hvilke type pakking som skal brukes og om innholdet skal distribueres til streaming og/eller webservere. For at dette skal være mulig så trengs en webservice på uploadserveren som kalles av klient applikasjonen etter at innholdet er uploadet. Webservicen pakker da innholdet i henhold til parameter den mottar og sender det videre til distribusjonsservicen. Se vedlegg D for skjermbilder fra uploadingsklienten som ble utviklet som en del av denne oppgaven.

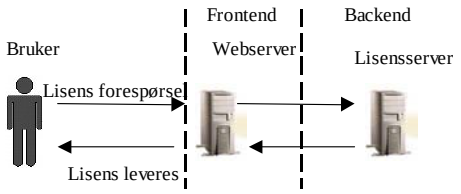


Figur 10: Upload alternativ 2, tilpasset uploading klient.

1. Filen lastes opp til upload og pakkeserveren ved hjelp av FTP.
2. Klienten sender parametere til webservicen som spesifiserer rettighetene som lisensene til filen skal inneholde, hvordan filen skal pakkes og hvor den skal distribueres. Webservice'en pakker deretter innholdet med de gitte parametrene. Distribueringen av innholdet til streaming og/eller webservice gjøres av samme distribusjonservice som for alternativ en.

3.4 Oppsett av lisensserver

Lisensserveren kjører på Internet Information Server versjon 5 og kjører en webservice som generer de ulike lisenstypene. Webservicen leverer lisenser for batch pakket innhold og Pay-Per-View innhold både som predelivery og som onDemand lisenser. Lisensene leveres brukeren gjennom webserveren. Arkitekturmessing så står lisensserveren i backend.



Figur 11: Viser en lisensforespørsel for en onDemand lisens.

Brukeren sender en lisens forespørsel til webserveren som kontakter lisensserveren for å få tak i en lisens til en fil. Lisensserveren genererer lisensen og sender den til webserveren som leverer den til brukeren.

Som nevnt tidligere så finnes det ulike metoder for å levere lisenser til brukere enten ved predelivery, silent eller nonsilent levering. Vi skal se nærmere på hvordan en mulig implementasjon kan se ut. Dersom en baserer seg på predelivery lisenslevering må en også ha silent eller nonsilent levering som en reserveløsning. Lisensserveren tilbyr en webservice som kan generere de ulike lisenstypene.

3.5 Oppsett av streamingserver

Streamingserveren som benyttes er Windows Media Services 9 Series som er en del av Windows .NET Server 2003 Enterprise Edition.

3.6 Distribusjonsalternativer

3.6.1 Bakgrunn

Etter at innhold er pakket distribueres det til streamingserverne og/eller webserverne automatisk. Distribusjonen kan gjøres enten ved å kopiere de beskytta filene over til serverne med en Windows service eller ved å overføre filene med FTP ved hjelp av en Windows service.

3.6.2 Distribusjon av innhold til streaming eller webservere på intern nett

Etter at innholdet er kryptert må det plasseres på streamingservere og/eller webservere for at brukere skal kunne laste det ned. For å automatisere distribusjonen av innholdet bruker jeg også her en Windows service. Service'en fungerer på samme måte som pakke servicen. Den sjekker ved gitte tidsintervaller om det finnes filer i bestemte kataloger, den vil da sjekke katalogene hvor pakke prosessen plasserer de beskytta filene. Den enkleste måten er da å benytte en Windows service til å kopiere disse filene over til streaming serverne og webserverne. Overføringen skjer ved at servicen benytter vanlig filkopiering for å kopiere de DRM beskytta filene over til streamingserverne. En forutsetning for at dette oppsettet skal fungere er at streamingserverne og pakke serveren står på samme lokalnett. Fordelen er at båndbredden på lokalnettet er mye høyere enn det en kan oppnå over Internett så distribusjonen av innholdet vil være mer effektiv en for distribusjon med FTP.

3.6.3 Distribusjon av innhold til streaming eller webservere med FTP

Distribusjonen av innhold kan utvides ved å distribuere innholdet ved hjelp av FTP protokollen. Fordelen er at pakkeserveren kan distribuere innhold til servere som står rundt om på Internett. Pakkeserver og streamingserver trenger da ikke stå på samme lokalnett som i forrige eksempel. Dette gir økt fleksibilitet i arkitekturen av systemet. Distribusjonen av innholdet setter store krav til FTP støtten som brukes av systemet. Det skal være mulig å overføre store kvantum med filer og størrelsen på filene kan overstige flere gigabytes.

Som et eksempel så kan vi finne ut hvor stor en fil vil være dersom det benyttes en bitrate på 950 Kbits/sekund eller 2000 Kbit/s, filmen har en lengde på to timer. Størrelsen på filene kan regnes ut med følgende formel:

$$(X\ Kbps * S\ seconds) / 8192 = Y\ MB \qquad [15]$$

Formelen gir et overslag over størrelsen, men det er tre faktorer som spiller inn på størrelsen til filen: bitraten som benyttes i koding av filen, lengden av innholdet og hvilke type innholdet det er. Dersom innholdet inneholder mye bevegelser i bilder vil det kreve mer enn innhold med mindre bevegelse i bildet.

Eksempel 1: Hvordan regne ut overslag over filstørrelser for WM9.

Bitrate → X = 950 Kbits/sekund
Filmens lengde → 2 timer = 7200 sekunder

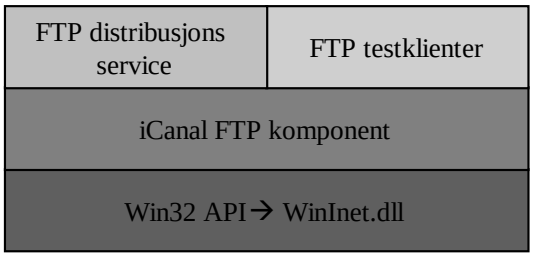
(950 Kbps * 7200 sekunder) / 8192 = 835 MB

Dersom en ønsker tilnærmet DVD kvalitet på filmen:
Bitrate → X = 2000 Kbit/sekund
Filmens lengde → 2 timer = 7200 sekunder

(2000 Kbps * 7200 sekunder) / 8192 = 1758 MB

Det finnes .NET klassebiblioteker som en kan laste ned gratis fra tredjepartsleverandører, men kvaliteten på noen disse er ikke alltid tilfredstillende nok eller så er de ikke gratis. For dette systemet trenger vi FTP støtte som fungerer stabilt ved overførsel av store filer som kan inneholde flere gigabytes med data. For FTP støtte er det valgt å bruke en standard Win32 komponent. Komponenten er WinInet.dll som leveres sammen med Internett Explorer fra Microsoft. Microsoft anbefaler at en bruker versjon 6.0.2600.0 eller nyere av WinInet.dll som distribueres sammen med Internett Explorer 6 servicepack 1.

Windows Internett komponenten tilbyr et API som støtter en rekke protokoller, blant annet FTP, HTTP og Gopher. WinInet komponenten gir applikasjoner mulighet til å navigere og manipulere kataloger og filer som ligger på en FTP server. Komponenten brukes i denne sammenhengen til å laste opp filer som er krypterte fra pakkeserveren til streamingserverne og/eller webserverne. Pakkingen av innholdet fungerer på vanlig måte, det er kun distribusjonsmetoden som er forskjellig fra metoden beskrevet i kapitel 3.6.2



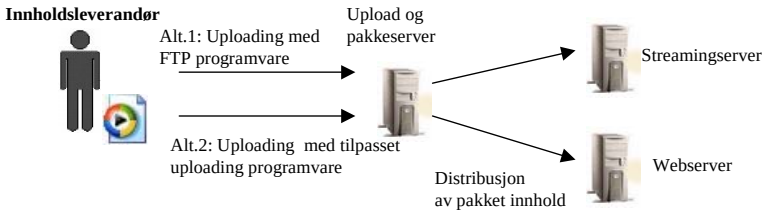
Figur 12: Arkitekturen til FTP distribusjonskomponenten.

Figur forklaring:

- Lag 1 består av WinInet komponeten fra Microsoft.
- Lag 2 består av en iCanal FTP komponent som ble laget for denne prototyp applikasjonen. Komponenten inneholder metoder for å utføre ulike FTP kommandoer. Hensikten med komponenten er å oversette formatet på parametrene fra .NET format til formatet som kreves av WinInet komponenten. Komponenten inneholder også en del metoder som kaller flere WinInet metoder i samme metode. Det letter programmeringen mot denne komponenten. For eksempel så inneholder komponenten en funksjon for å gjøre en FTP put kommando. Icanal FTP komponenten sørger da selv for å sette opp forbindelsen og lukke denne etterpå.
- Lag 3 består av klienter som benytter iCanal FTP komponenten. Hovedsakelig er dette FTP distribusjonsservicen som distribuerer innholdet fra upload/pakke serveren til streamingservere og/eller webservere. Komponenten brukes også av ulike test klienter til ulik testing av DRM systemet.

3.6.4 Oversikt over hvordan komponentene spiller sammen

Figuren under viser hvordan de ulike komponentene beskrevet i dette kapittelet spiller sammen.



Figur 13: Oversikt over de automatiserte publiseringsalternativene.

Som en ser av figuren over så kan innhold enten lastes opp med FTP eller en tilpasset uploadingsklient. Selve pakkingen av innholdet er litt forskjellig for de to alternativene. For alternativ 1 pakkes innholdet med en Windows service, mens for alternativ 2 pakkes innholdet med en webservice etter at innholdet er lastet opp.

Distribusjonen av innholdet til webservere eller streamingservere gjøres for begge alternativene med en Windows service som laster opp innholdet til streaming og eller webserverne. Webservicen som brukes til pakking av innhold og Windows servicene for pakking og distribusjon kjører alle på upload/pakkeserveren.

4 Design av prototyp

4.1 Design av database

De to forskjellige alternativene for pakking av innhold benytter databasen noe forskjellig. For normal pakking av innhold genereres det nye verdier for keyid, nøkkelpar og seed for hver fil. Verdiene lagres i en database slik at de kan benyttes i lisensgenereringen. Men under batchpakking av innhold benytter pakkeprosessen verdier for keyid, nøkkel par og seed som finnes i databasen. Pakkeprosessen for batch pakking finner verdier i databasen som er assosiert med innholdsleverandøren som uploadet innholdet.

Databasen inneholder også informasjon om rettigheter assosiert med en keyid. Når lisensserveren genererer lisensene finner den hvilke rettigheter som skal settes i lisensen i databasen.

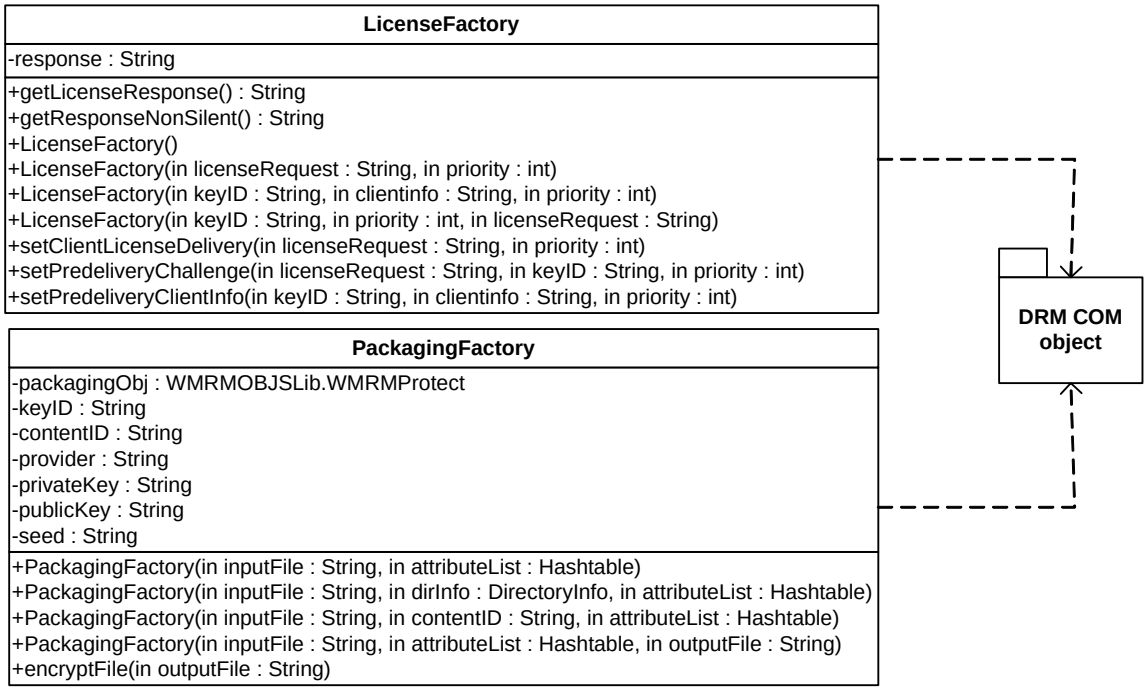
Figur 14: Datamodell over DRM data

ContentIDregistery tabellen inneholder verdier som benyttes for batchpakking av innhold. Pakkeprosessen finner ut hvem (ut ifra hvilke katalog det er uploadet til) som har uploadet innholdet og henter keyid og nøkkel par og seed i databasen på grunnlag av dette.

Når innholdet pakkes vil det legges inn data om innholdet i LicenseData tabellen. Dette gjøres både for batch og normal pakking av innhold. LicenseData tabellen inneholder med dette all informasjon som trengs for å generere lisenser til innholdet.

4.2 Design av DRM rammeverket

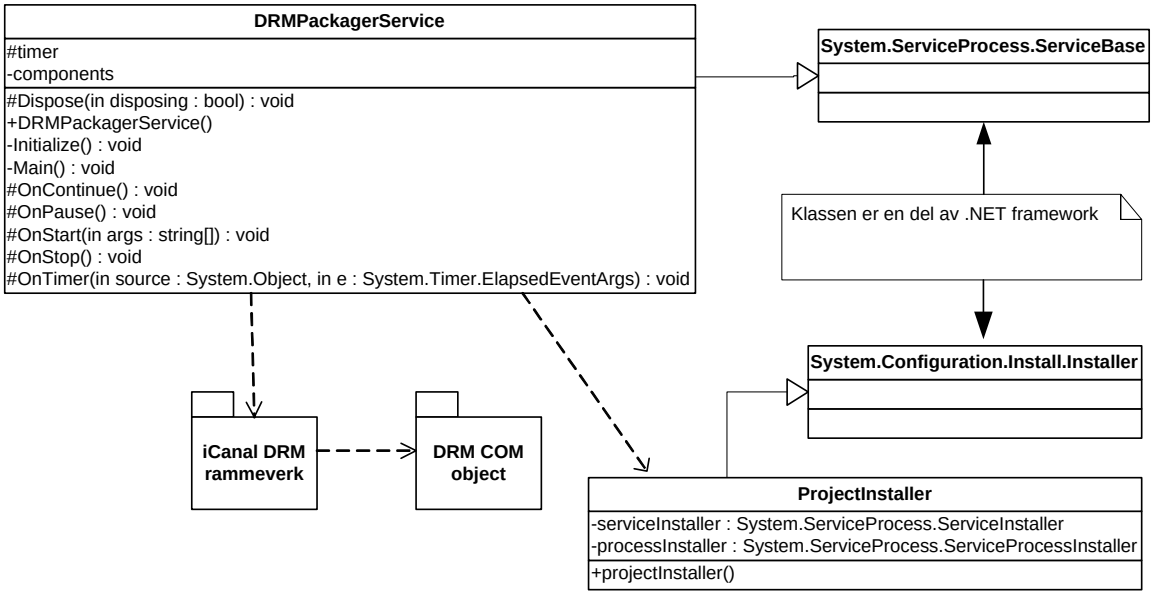
Figuren under viser et forenkelt klassediagram over iCanal DRM rammeverket. Klassediagrammet viser strukturen til factoryklassene. Klassene benyttes av lisensserveren og pakkeserveren til å pakke og levere lisenser til innhold. Factoryklassene har metoder for å generere alle typer lisenser og pakke innhold for Pay-Per-View eller abonnement.



Figur 15: Forenklet klassediagram for iCanal DRM rammeverket.

4.3 Design av pakkeservice og distribueringsservice

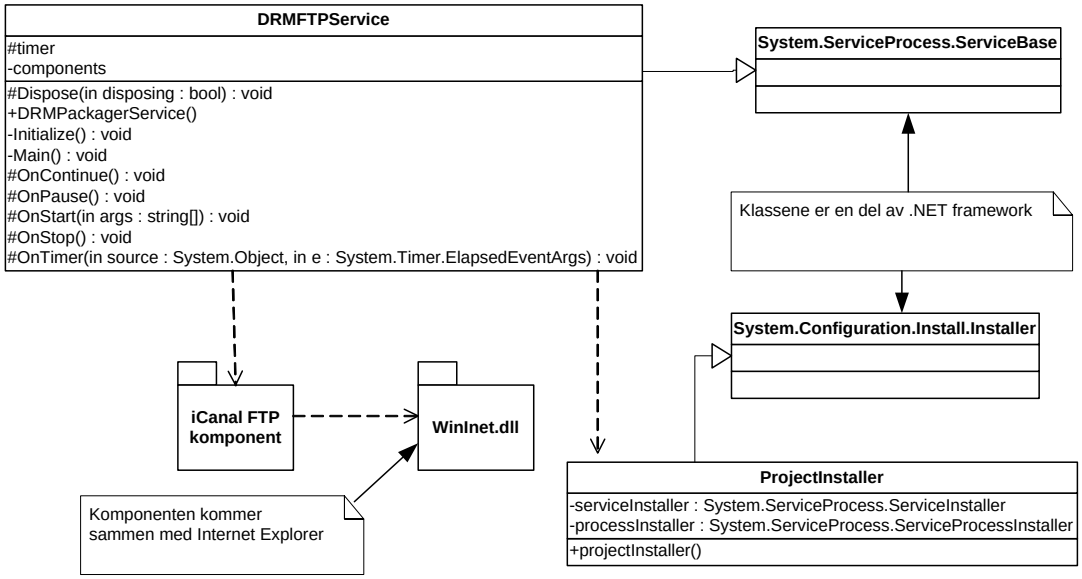
Designet av servicene for pakking og distribusjon av innholdet er rimelig likt. Det som skiller de to er innholdet i OnTimer funksjonen. OnTimer trigges ved gitte tidsintervaller. Pakkeservicen pakker nytt innhold som er uploadet og distribuseringsservicen distribuerer innholdet som er ferdig pakket til streaming og/eller webserveren.



Figur 16: Klassediagram for pakkeservice'en.

Pakkeservicen benytter iCanal DRM rammeverket til å pakke innholdet som er uploadet til uploadserveren.

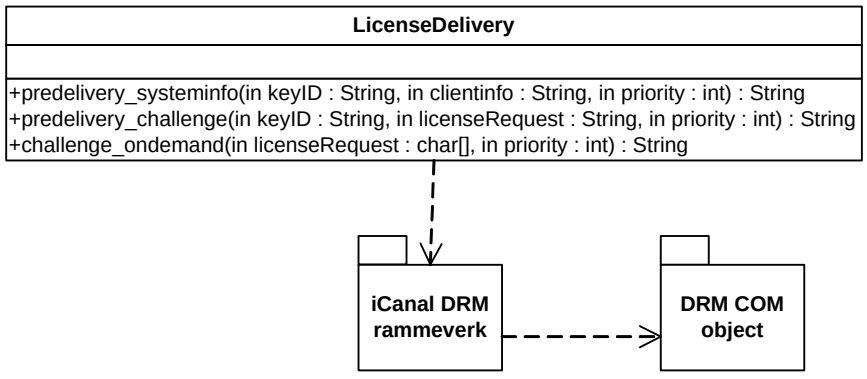
Distribusjonsservicen benytter iCanal FTP komponenten for å distribuere innholdet til streamingservere og/eller webservere.



Figur 17: Klassediagram for distribusjonsservice'en.

4.4 Design av lisensserver

Lisensserveren som er utviklet kan generere lisensene på tre forskjellige måter. Den kan generere to typer predelivery lisenser enten basert på en licenseRequest eller ved en systeminfo request. LicenseRequesten genereres av Mediaplayeren når den finner ut at den mangler lisens for innholdet den skal spille av. Mens systeminfo genereres ved hjelp av netobj.getclientinfo() funksjonen.



Figur 18: Klassediagram for lisensserveren.

- Predelivery_systeminfo(…) – generer lisenser for predelivery basert på systeminfo fra klienten og mediafilens keyid.

- Predelivery_challenge(…) – generer lisenser for predelivery basert på en challenge fra klienten og mediafilens keyid.
- Challenge_ondemand – generer ondemand lisenser basert en licenseRequest fra klienten.

Prioritet parameteret kan benyttes for å sette ulike prioriteringer på lisenser til samme fil. Lisensen med høyest prioritet vil da brukes ved avspilling av mediafilen.

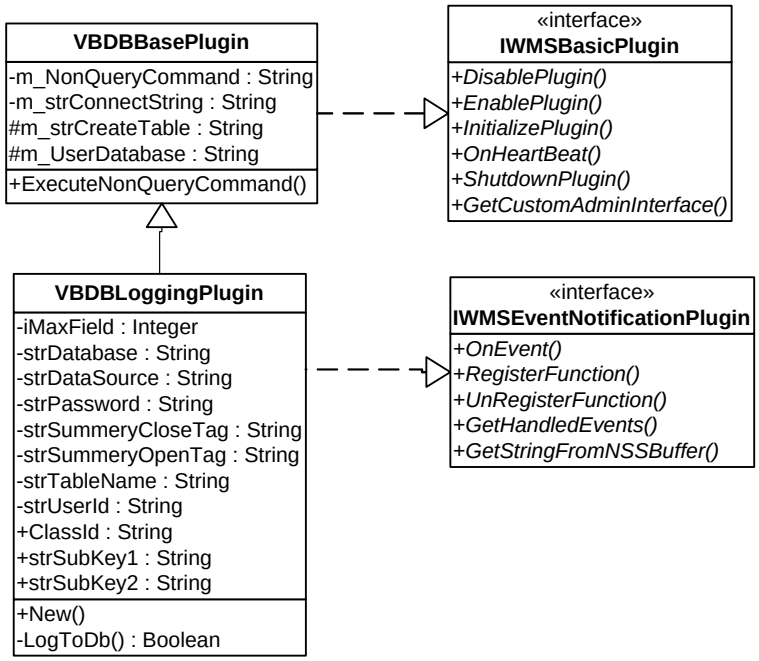
4.5 Design av logging plugin til streamingserveren

I kommersielle streamingløsninger er logging og rapporteringen en sentral del. Logger fra streamingserveren kan for eksempel benyttes til analyse av historiskedata. Løsninger kan ha forskjellige krav til lagring av historiske data for streamingserveren både når det gjelder hvilke innhold som streames, klienter som brukes eller hvor mye innhold som streames fra de ulike publishingpoint'ene. For løsninger hvor en baserer seg delvis eller helt på reklameinntekter kan det tenkes at en har en Pay-per-View modell for reklameinntektene. I dette tilfellet er det viktig å logge hvilke og hvor mange ganger en reklamefilm er vist på siten. Andre scenarioer kan være at en aksessleverandør ønsker å analysere hvilke innholdet som er mest populært osv.

Windows Media Services 9 Series baseres på en plugin arkitektur. Streamingserveren er laget slik at en enkelt kan lage egendefinerte plugins som passer den enkelte organisasjon. Plug-ins'ene skrives i .NET, C++, C# eller VB.NET, og bruker COM teknologi, noe som gir god fleksibilitet for utviklere som ønsker å tilpasse løsninger [16]. Plugins kan også aktiveres på forskjellige nivåer på serveren. For eksempel kan en aktivere en plugin på server nivå, da vil plugin'en kjøres på alle publishing points for serveren. Plugins kan også aktiveres på publishing points nivå det vil si at en kan aktivere en bestemt plugin for ett eller flere publishing points. Det lages egne instanser av pluginen for de forskjellige nivåene, noe som gjør det enkelt å konfigurere plugins forskjellig for forskjellige publishingpoints. En kan gjøre dette gjennom å lage et administrator interface hvor en kan sette properties.

Media Services tilbyr et vidt spekter av interface'er som en kan integrere mot. Det gjør det enkelt for programmere å komme raskt i gang med utviklingen. Mange av interface'ene er bygget rundt en event modell. Det vil si at serveren trigger ulike funksjoner på forskjellige eventer fra serveren. Serveren sender ut meldinger som inneholder ulik informasjon og en kan ved implementasjon av plugins bestemme hvordan en skal behandle de ulike meldingene.

Det er laget en plugin i prototypen som logger aktiviteten på streamingserveren. Dataene logges i en database og kan benyttes til ulike analyse formål.



Figur 19: Klassediagram for logging plugin'en

Når en skal lage plugins for logging er det to interface'er en må bruke: IWMSBasicPlugin og IWMSEventNotificationPlugin[17]. Interface'ene gir programmeren tilgang til å programmere mot Media Serveren.

IWMSBasicPlugin innholder metoder for å initiere, aktivere, deaktivere, avslutte, intervall trigging og gi tilgang til et administrator interface for plugin'en. Interface'et anbefales å implementeres for alle egenproduserte plugins.

IWMSEventNotificationPlugin inneholder metoder for å behandle event meldinger som kommer fra serveren. Ved å bruke disse to interfacene kan en enkelt få tilgang aktivere, slå av, administrere eller behandle meldinger fra Media serveren.

5 Implementasjon av prototyp

5.1 Bakgrunn

Implementasjonen av prototypen ble gjort i .NET utviklingsmiljøet. De ulike delene av prototypen ble utviklet i C#, bortsett fra streamingserver plugin'en som ble skrevet i VB.NET.

Som vedlegg A ligger kildekoden til alle komponentene som er utviklet i denne oppgaven: DRM rammeverket, pakkeservicen, distribusjonsservice for intern nett, distribusjonsservice for Internett (FTP), tilpasset uploadingsklient, PackagingWebService, lisensserver og Web applikasjon.

E-handelssystemer inneholder en produktkatalog med alle produktene en organisasjon tilbyr. For eksempel kan produktkatalogen inneholde en oversikt over Pay-Per-View produkter og ulike abonnement på innhold. Dersom en skal benytte predelivery av lisenser må det finnes en mapping mellom produktkatalogen og lisensinformasjonen som ble lagret i databasen under pakkeprosessen. Prototypen tar ikke for seg hvordan en mapper lisensinformasjon og produktkatalogen, men produktkatalogen må inneholde en nøkkel som kan brukes til å finne frem til riktig lisensinformasjon i lisensdatabasen. En måte å gjøre dette på er å benytte keyid'en som nøkkel mellom produkt og lisensdatabasen eller en kan generere en egen nøkkel. Dersom en benytter onDemand lisensiering kan en klare seg uten denne mappingen. Mediaplayeren finner da selv ut hvor en får tak i lisens til filen og keyid'en er som kjent registrert i headeren til filen.

5.2 Implementasjon av FTP distribusjonsservice

Implementeringen av WinInet og .NET teknologien bøy på en del utfordringer. Siden WinInet ikke er skrevet i .NET kan en ikke stole fullt ut på API dokumentasjonen for komponenten. API dokumentasjonen er skrevet for C++ og VB og blir noe forskjellig for .NET. Flere av funksjonssignaturene for WinInet inneholder attributter og returverdier av typen "long". VB versjon 6 og VB.NET/C# "long" verdier representeres av ulikt antall bytes. En long variabel i VB versjon 6 er 4 bytes, mens en long i .NET er 8 bytes. For at implementeringen skal fungere i en .NET applikasjon må en representere longverdiene som .NET integer verdier. Integer verdier representeres som 4 bytes verdier. ICanal FTP komponenten er laget for å ta hensyn til disse programmeringsspråk forskjellene, den inneholder derfor metoder som gjør mappingen mellom verdiene som skaper problemene.

5.3 Klient PC kompabilitets test

Erfaringer fra tidligere prosjekter som iCanal har gjennomført og tilbakemeldinger fra ulike samarbeidspartnere viser at det er viktig at en tjeneste er enkle å bruke for sluttbrukeren. Det hjelper lite å publisere og distribuere innholdet til brukeren dersom han ikke skjønner hvordan det skal brukes eller hva som trengs for å benytte innholdet. Det er viktig å presisere for brukeren hva som kreves av tjenesten både når det gjelder hardware og software krav. Det hjelper lite å ha et bra VoD system på serversiden dersom brukeren ikke skjønner hvordan det skal brukes. Dersom software krav ikke er tilfredstilt bør det være enkelt for brukeren å oppdatere software slik at kravene tilfredstilles. For eksempel dersom brukeren ikke har Windows Media playeren installert må brukeren underrettes om det og få presentert en enkel oversikt over hvordan han kan oppfylle kravene. Under følger en oversikt over hardware og software krav som bør oppfylles dersom innholdet skal kunne spilles av hos brukeren.

5.3.1 Hardware

Når det gjelder krav til en Video on Demand tjeneste så vil dette kreve en del av både hardware og software. Video krever en del av maskinvare får å gi brukeren et bra resultat. Opplevelsen av en VoD tjeneste blir ikke noe særlig dersom filmer skulle "hakke", derfor er det viktig at brukeren har maskinvare slik at dette unngås.

Tabell 2: Anbefalte krav for å kjøre VoD. Kilde [6] og [18]

Komponent	Anbefalte krav
Prosessor:	Pentium III 733 MHz (eller tilsvarende)
Internminne:	128 MB RAM (eller bedre)
Grafikkort:	16 MB (eller bedre)

5.3.2 Software

Det kreves en del software for at denne VoD løsningen med DRM skal fungere. Arbeidet er konsentrert rundt Windows Media 9 Series produkt familien, noe som setter krav til software som trengs. Windows Media Playeren kjører på Windows og Windows Media 9 kodek brukes.

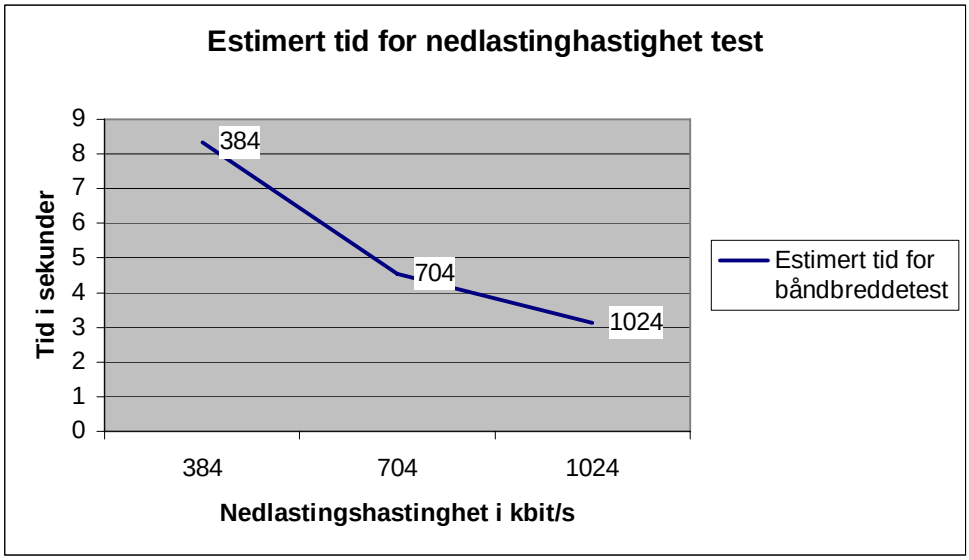
Tabell 3: Software krav brukeren må oppfylle for å kunne benytte VoD tjenesten.

Komponent	Anbefalte krav
Operativ system	Windows 98 SE/ME/2000/XP
Nettleser	Explorer 6.0 (eller nyere)
Player	Windows Media Player 9

For at det skal være enkelt for brukere å benytte tjenesten så er det i dette arbeidet laget et forslag til hva og hvordan en kan teste klient PC'en for at VoD tjenesten skal fungere tilfredstillende. Alle testene av klient PC'en gjøres på klient PC'en med Javascript. Det er laget Javascript rutiner som tester diverse software på klient PC'en og måler nedlastingshastigheten en bruker har tilgang til. Javascript rutinene er integrert i en HTML side så testene av systemet gjøres gjennom denne siden. Se vedlegg D for skjermbildet av siden. Kildekode for Javascript rutinene finner en på CD vedlegget, vedlegg A. Dersom det mangler komponenter på klient PC'en vil brukeren bli bedt om å installere disse før VoD tjenesten startes.

5.3.3 Nedlastingshastighet test

Siden VoD tjenesten kun er rettet mot bredbåndsmarkedet så er det viktig at en tester båndbredden til brukere som bruker tjenesten. Brukere som ikke sitter på en bredbåndsforbindelse vil ikke ha mulighet til å kjøre VoD tjenesten. Testen for å teste båndbredden er enkel å implementere og den gir rimelig bra måleresultater. Det gjøres ved å måle tiden det tar å laste ned en fil over internettforbindelsen til brukeren. Filen som brukes i målingen er 400kB stor noe som gir gode gjennomsnittsmåleresultater. Det er fullt mulig å bruke testfiler som er mindre i størrelse, men større feilmarginer må da påregnes. Større testfiler vil gi enda bedre gjennomsnittsmålinger, men testen vil da ta lenger til å gjennomføre.



Figur 20: Estimert tid for båndbredde testen for ulike bredbåndsalternativer. Størrelse på test filen er 400 KB

Windows testen
Det testes at klient PC'en er en Windows maskin, DRM fungerer per i dag ikke på Linux plattformer blant annet. DRM for Linux er et omdiskutert tema for tiden, en av utfordringene en står overfor her er hvordan en kan lage en DRM løsning på en opensource plattform.

Internet Explorer 6 eller nyere
Det kreves at brukere har Internet Explorer 6 eller nyere på grunn av at det benyttes iFrames i blant annet systemtesten. iFrames støttes i IE versjon 6 og nyere.

Tillatt IP- adresse.
Sjekker at brukeren sitter på en gitt IP range, brukes til å sjekke at brukeren som prøver å aksessere tjenesten sitter på nettet til distributøren.

Cookies test
Cookies må kunne skrives til klient PC'en på grunn av at det lagres ulike cookies på klient PC'en som blant annet inneholder informasjon om klient PC'en.

Windows Media test.
For at brukeren skal ha best mulig utbytte av VoD tjenesten bør en ha versjon 9 av Mediaplayeren, men løsningen fungerer også med versjon 6.4 eller 7 av playeren hvis 9 series kodekene er installert.

Individualisert player test
Individualisering er viktig med hensyn på sikkerhet for VoD systemet. Ved å individualisere playeren gjøres playeren unik ved å linke den til den enkelte datamaskinen. Det forhindrer at playere som har blitt hacket ikke kan distribueres i stor skala over Internett. Dersom en player hackes vil det kun gjelde for den ene maskinen. Med individualisering kan en identifisere om playere er individualisert før en deler ut lisenser. Dette øker den totale sikkerheten for DRM systemet. Individualiseringen gjøres ved hjelp av Windows Media Format SDK'en. Som nevnt tidligere er dette den delen av DRM systemet til Microsoft som ligger på klientsiden i systemet, delen av systemet som er integrert i klient player applikasjonen. Første gang en initierer DRM komponenten i playeren vil playeren bli gitt en unik id noe som gjør at alle playere blir ulike. Siden individualisering øker sikkerheten anbefales det å ha med dette i integrasjonen av DRM systemet. Playeren individualiseres ved hjelp av Javascript som initierer DRM støtten i playeren.

Klokke test
Testen er viktig for at DRM systemet skal fungere best mulig. Det er viktig at klokkene på lisensserveren og klient PC'en er noenlunde synkrone på grunn av tidsbegrensingene som kan settes i lisensene. En kan tenke seg følgende senario, hvis lisensserveren sin klokke er 23 timer før klokken på klient PC'en så vil en lisens som i utgangspunktet skal være gyldig i 24 timer kun være gyldig i en time. En bør i et VoD system med DRM alltid sette rettigheten "DeleteOnClockRollback" eller "DisableOnClockRollback" til sant. En bør gjøre det for å unngå svindel med rettighetene, det vil si at en bruker skaffer seg urettmessig tilgang til innholdet ved å stille tilbake klokka på PC'en sin. Dersom DeleteOnClockRollback" eller "DisableOnClockRollback" rettighetene til lisensen er satt til sant vil lisensen enten slettes eller ikke fungere på klient PC'en dersom klokken stilles tilbake. (For den nye versjon 9 av SDK'en er ikke denne synkroniseringen like viktig fordi det her finnes en ny rettighet som kan settes til antall timer til lisensen utgår.)

5.4 Logging og rapportering fra streamingservere

5.4.1 Log typer

Windows Media Services 9 støtter alle versjonene av Windows Media Player. De forskjellige versjonene av Windows Media Player bruker forskjellige protokoller for å kontakte Media Serveren. Windows Media Player 9 series bruker HTTP/1.1 og Real Time Streaming Protocol (RTSP), mens de eldre versjonene av Windows Media Playeren bruker HTTP/1.0 og Microsoft Media Server (MMS) protokollene. Loggene sendes i forskjellige formater basert på hvilke protokoll som brukes. For HTTP/1.0, HTTP/1.1 og RTSP sendes logdataene i XML format fra klienten til serveren, mens for MMS så sendes logdataene i et binært format.

En kan dele loggingen i forskjellige kategorier:

- klient logger
    - client renders logs – blir sendt automatisk av playere som er basert på Windows Media Format SDK. For unicast sendes loggene til Windows Media Serveren og for multicast sendes loggene til en alternativ log server. Det kan gjerne være en webserver. Denne typen log sendes serveren når playeren skifter tilstand, det vil si når playeren starter/stopper, spoler osv.
    - client stream logs – loggen inneholder info om hvordan klienten mottok data, men ikke hvordan det ble renderet.
    - combination logs – brukes for player versjoner tidligere enn 9 series, men brukes ikke for fast cache, de to første logtypene er disablet på

serveren eller scenarioer hvor fast cashe ikke støttes for eksempel live stream.

- server logger – nye i 9 series.
  - Distribution logs – brukes av distribusjonsservere.
  - Propagated cache/proxy logs – brukes i tilfeller hvor en har en cache/proxy server f.eks i et intranett.

En nyhet i 9 Series er at playeren nå sender to typer logger til serveren det er client renders logs og client stream logs, i tidligere versjoner så ble det sendt en log som var en kombinasjon av disse to. I 9 Series sendes loggene som to typer pågrunn av Fast Cache mekanismen, som nevnt tidligere så betyr det at innholdet caches lokalt hos brukeren raskere enn real-time hastigheten. Derfor kan nedlastingen avsluttes før hele filen er avspilt.

5.4.2 Logging plugin

Avsnittet beskriver mer i detalj hvordan en kan implementere en plugin som behandler loggdata fra Media serveren og hvordan dataene kan brukes som rapporteringsmateriale. Pluginen behandler data fra streamingserver og lagrer dataene i en database.

Når logging pluginen aktiveres for et publishing point så registerer den seg i streaming serveren slik at den mottar meldinger fra serveren. Serveren sender meldinger til pluginen ved å trigge OnEvent() funksjonen i plugin'en. Pluginen får da meldingen som et input parameter og er i stand til behandle den. Siden meldingene er basert på XML formatet er det enkelt å finne feltene en er ute etter siden disse er lagret under respektive noder i XML DTD'en.

Pluginen som er laget i dette arbeidet behandler meldingene fra streamingserveren og logger dataene til en database. Dataene som logges i databasen kan da behandles med ulike rappoteringsverktøyer for å trekke ut informasjon som kan være nyttig. Meldingene fra Media serveren leveres i XML format noe som gjør det enkelt å få tak i den informasjonen en måtte ønske. Summery taggen inneholder all informasjon, men det enkleste er å aksessere informasjonen gjennom XML taggene. Ved å bruke XML format i meldingene så trenger en ikke lenger å følge W3C standarden som blant annet ikke støttet mellomrom i verdiene. XML strukturen er fleksibel slik at en enkelt kan legge til egendefinerte felt.

```
<XML>
  <Summary>0.0.0.0 2002-01-16 01:43:45 …</Summary>
  <c-ip>...</c-ip>
  <date>...</date>
     ...
    <s-proxied>...</s-proxied>

  <ContentDescription>
    <field1>...</field1>
    <field2>...</field2>
    ...
  </ContentDescription>

  <Some3rdPartyNamespace>
    <field1>...</field1>
    <field2>...</field2>
    ...
  </Some3rdPartyNamespace >
     …
</XML>
```

Figur 21: Eksempel på meldingsformatet fra Media serveren. Se vedlegg B for eksempel på en fullstendig melding.

For eksempel dersom en ønsker å knytte logdataene for hver fil til innholdsleverandøren som har levert filen kan en definere felter for dette i logdataene. Det blir da enkelt å trekke ut logdata som tilhører de forskjellige innholdsleverandørene.

5.5 Packaging WebService

Webservicen pakker innholdet som blir uploadet med den tilpassa uploadingsklienten. For at webservicen skal kunne pakke store filer må en øke timeout intervallet på webseriven for at den ikke skal gå i timeout på grunn av at pakkingen tar lang tid. Det gjør en ved å sette timeout på proxyklassen.

```
this.Timeout = 1800000; // 30 minutter
```

5.6 Demonstasjonswebsider

Prototypen for VoD systemet inneholder ikke bare businesslogikk for pakking, distribusjon av innhold og lisensdistribusjon. Den inneholder også noen svært enkle websider som demonstrerer hvordan VoD/DRM teknologien vil se ut for brukeren. Som nevnt tidligere er det ikke laget en produktkatalog for systemet så sidene er bygget opp statisk. Produktkatalogens struktur og oppbygning var ikke en del av

denne oppgaven. Demonstrasjonssidene inneholder logikk for å dele ut lisenser og demonstrerer de ulike distribusjonsalternativene for innholdet: streaming, progressiv nedlasting osv. Se vedlegg D for skjermbilder fra demonstrasjonssidene.

6 Distribusjonsalternativer for innhold.

6.1 Bakgrunn

En del av oppgaven var å se på fordeler og ulemper ved ulike distribusjonsalternativer for innhold fra aksessleverandør til sluttbruker for et VoD system basert på Microsoft teknologi. Det vil si metoder for å overføre innholdet fra aksessleverandør til sluttbrukeren. Testene av de ulike distribusjonsalternativene ble utført på prototyp systemet for å teste systemet og for å avdekke fordeler og ulemper ved de ulike publiseringsalternativene. Innholdet ble testet for alle distribusjonsalternativene, brukervennlighet og hvordan de ulike alternativene fungerte funksjonelt ble dokumentert.

Når en skal evaluere distribusjonen av innhold kan en evaluere ut ifra forskjellige kriterier: videokvalitet brukeren kan oppnå ut ifra nedlastingshastigheten som er tilgjengelig for streaming, buffering tid eller nettverksøkonomisk sett fra aksessleverandør. Måling av videokvalitet er ikke en del av denne oppgaven men tall hentet fra Telenor og NextGenTel varierer litt. Tall fra Telenor viser at dersom en skal oppnå tilnærmet DVD kvalitet på streamingen må en kode innholdet med bitrate opp mot 2 Mbit/s, mens NextGenTel opererer med en bitrate på 1,5 Mbit/s for Windows Media 9 formatet. Dersom en har krevende innhold dvs med mye bevegelser og detaljer i bildet så må en opp mot 2 Mbit/s, mens innhold som er mindre krevende kan fungere bra med 1,5 Mbit/s. Dersom en ønsker videokvalitet opp mot dette trenger en nedlastingshastigheter som ligger en del høyere enn det som er vanlig for ADSL abonnementene i dag. Tallene hentet fra Telenor og NextGenTel viser med en gang at en ikke kan oppnå videokvalitet opp mot DVD kvalitet for streaming for de fleste bredbåndsabonnementene som finnes på markedet i dag. En metode for å dekke inn mangelen på båndbredde er å benytte nedlasting av innhold.

Innholdet kan distribueres til brukeren på forskjellige måter, en kan benytte streaming, nedlasting eller progressiv nedlasting av innholdet. De forskjellige distribusjonsmetodene har sine fordeler og ulemper. Arkitekturen som er utviklet i prototypen åpner for bruk av alle disse distribusjonsmetodene. Vi skal nå se resultatet av bruken av disse ut ifra brukerens og aksessleverandør ståsted når det gjelder onDemand innhold.

Hovedpoenget med testene som ble utført var å avdekke funksjonaliteten for de ulike distribusjonsalternativene og finne ut hvordan de ulike distribusjonsalternativene opererer i ulike scenarioer. Hvilke funksjonalitet finnes i

de ulike alternativene som er av interesse for bruker og aksessleverandør. Det er ikke lagt vekt på måling av nedlastingstider og måling av videokvalitet for de ulike løsningene. Måling av videokvalitet er dekket av tidligere arbeid utført av Telenor FOU. Og viser at videokvaliteten økes når en øker bitraten innholdet blir kodet med. Nedlastingstiden vil variere veldig ut ifra hvilken aksess brukeren har. Det er først og fremst funksjonalitet som testes i de ulike løsingene. Resultatene fra testene under er basert på litteraturstudier og praktisk testing av de ulike alternativene.

6.2 Streaming av innhold

6.2.1 Tradisjonell streaming

Fordelene med streaming er at brukeren kan starte å se innholdet nesten umiddelbart. Playeren starter avspillingen av innholdet etter at den har bufret innholdet i en kort periode. Buffering tiden er 5 sekunder som standard og maks 60 sekunder i Mediaplayeren. Brukeren trenger ikke vente til hele filen er lastet ned på maskinen før den kan benyttes. Distribusjonen av innholdet skjer i sanntid og innholdet blir ikke lagret på klient PC'en. Innholdet lagres kun i minnet på maskinen og skrives derfor ikke til disk. En stor fordel for brukeren er den korte buffering tiden før innholdet spilles av. Den praktiske testen viser at brukeren kan også spole frem og tilbake i filen. En bakdel er at videokvaliteten på innholdet begrenses av nedlastingshastigheten brukeren har tilgang til på grunn av innholdet overføres i tilnærmet sanntid. For eksempel dersom brukeren har en aksesslinje på 704Kbit/sek vil en typisk kode innholdt med en bitrate på 650Kbit/sek. En bør kode innholdet med en lavere bitrate enn nedlastigshastigheten for å være sikker på at aksessen håndterer nedlastingen av innholdet. Det er ikke mulig å stream innhold med en høyere bitrate som er høyere enn nedlastingshastigheten bruker har tilgang til. Streamingen vil da "hakke" siden playeren ikke klarer å laste ned alle dataene på grunn av begrensingene i nedlastingshastigheten.

Sett fra aksessleverandøren er dette et nettverksøkonomisk kostbart alternativ. For eksempel dersom en bruker har betalt for tilgang til en film i et døgn og velger å se den en eller flere ganger så må filmen overføres til brukeren hver gang. Men den nye streaming plattformen fra Microsoft har ny funksjonalitet, fast streaming, som kan bidra positivt for å løse noen av de negative sidene ved tradisjonell streaming.

6.2.2 Fast streaming

Fast streaming kommer med den nye 9 Series streaming plattformen fra Microsoft og omfatter funksjonalitet som forbedrer vanlig streaming betraktelig. Fast streaming består av et sett tilleggsfunksjoner som tradisjonell streaming ikke har:

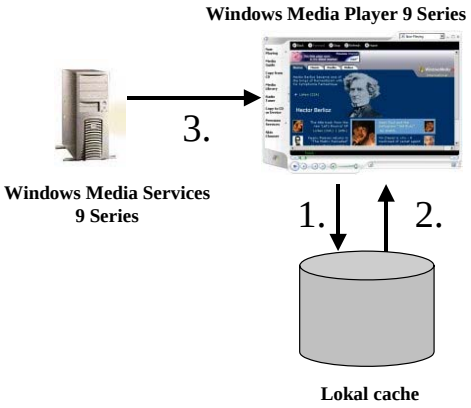
- Fast start – fyller opp bufferet så raskt som mulig ved hjelp av all tilgjengelig nedlastingshastighet. Bufferet fylles opp raskere enn ved vanlig streaming og fører til lavere buffering tid enn ved vanlig streaming.
- Fast cache – dersom det er ekstra båndbredde tilgjengelig vil dette brukes til å øke bufferet mens filen spilles av.
- Fast reconnect – Gjenoppretter server/player koblinger automatisk dersom de blir brutt.
- Fast recovery – feilkorreksjon av mottatte pakker.

Fast streaming fungerer stort sett som vanlig streaming, men innholdet kan caches lokalt hos brukeren i et gitt periode. En kan sette hvor lenge innhold som streames fra et publishing point skal caches hos brukeren. Sett fra aksessleverandør er denne funksjonaliteten interessant fordi den kan være med på å spare streamingkostnader og reduksjon i nettverkstrafikken. Overføringen av data kan også overføres raskere en det som trengs i renderingen noe som skiller seg litt fra vanlig streaming hvor overføringshastigheten bestemmes av avspillingsbitraten. Fast cache er kun støttet i Windows Media Player 9 Series og Windows Media Format 9 Series SDK. Fast cache går ut på å laste ned innhold raske enn hastigheten innholdet spilles av. Det sier seg selv at fast cache kun støttes for onDemand innhold og ikke i live-stream scenarioer. (live-stream er sanntidsinnhold) Med fast cache vokser bufferet dersom en har høyere nedlastingshastighet tilgjengelig enn bitraten innholdet er kodet med. I dette tilfelle vil en være bedre rustet mot variasjoner i nedlastingshastigheten enn ved vanlig streaming, siden bufferet med fast cashe vil være større enn for vanlig streaming. Den praktiske testen viser at spole funksjonaliteten fungerer ubegrenset også for dette alternativet.

Fast streaming fungerer slik at når brukere ønsker å laste ned innhold enten fra en fil eller fra en playliste i et publishing point så vil playeren først sjekke den lokale cachen for å se om innholdet allerede er lastet ned. En kan bruke plugins på publishingpoints for å sette hvor lenge innhold skal beholdes i cachen til brukeren før det slettes. Hvis innholdet finnes i den lokale cachen og fortsatt er gyldig ut ifra expire parameteret som ble satt i publishing pointet vil innholdet spilles av lokalt. Det er veldig praktisk sett fra en ”nettverksøkonomisk synsvinkel” siden brukeren ikke trenger å laste ned innholdet enda en gang. Når innhold spilles av lokalt vil playeren sende rendering logs til Media Serveren. (også kalt ”Player-Cache Hit scenario”). Dersom playeren ikke finner innholdet i den lokale cachen så vil playeren streame og cache innholdet fra Media serveren. (kalt Player-Cache Miss scenario) I dette tilfellet vil playeren sende både streaming og rendering logger til Media serveren.

Det finnes noen faktorer som bestemmer om fast streaming kan brukes eller ikke:

- Klienten må ha tilgang til høyere båndbredde enn det som kreves for får å spille av innholdet i normal hastighet. Den ekstra båndbredden brukes da til å cache innhold lokalt hos klienten.
- Hvis brukeren ikke har mer enn 100 MB ledig på harddisken så vil innhold ikke caches. Dersom ledig diskplass ikke er mer enn 100MB vil playeren gå over til vanlig streaming, dvs. at innholdet ikke caches.
- Dersom playeren spiller en playlist vil playeren kun streame elementer som ikke finnes i cachen. Finnes elementet i playlisten i cache vil playeren spille innholdet lokalt.



Figur 22: Oversikt over fast cache funksjonaliteten.

Som en kan se av figuren over så sjekker playeren om innholdet finnes i den lokale cachen(punkt 1). Dersom innholdet finnes lokalt så lastes innholdet lokalt (punkt 2), hvis innholdet ikke finnes lokalt må det hentes fra streamingserveren (punkt 3).

Det ble utført noen pratiske tester for dette distribusjonsalternativet. Testene viste at fast cache funksjonaliteten fungerte. Brukeren må være online for å starte avspillingen av cachet innhold, men etter at streamen er startet kan brukeren gå offline dersom innholdet er cachet lokalt. Tester viste også at dette alternativet er mer robust mot variasjoner i nedlastingshastighet enn vanlig streaming. Dersom playeren har rukket å cache en del data vil bufferet være betydelig større siden det ikke er noen fast bufferstørrelse som ved tradisjonell streaming hvor buffer størrelsen er fast. Testen ble utført ved at jeg spilte av innholdet like lenge, for eksempel 5 minutt, for både tradisjonell streaming og fast streaming. Etter 5 minutt belastet jeg aksessenlinjen med 40% ekstra trafikk fra andre applikasjoner, for tradisjonell streaming så gikk bufferet tomt rimelig raskt. For fast streaming fortsatte avspillingen en del lenger på grunn av at bufferet var større for fast

streaming enn for tradisjonell streaming. Testen ble gjennomført på en 704 Kbit/sek ADSL linje med en fil med bitrate på 650 Kbit/sek.

6.3 Nedlasting av innhold

6.3.1 Normal fil nedlasting av innhold

Nedlasting av innhold er positivt dersom en ønsker å fokusere på videokvalitet dersom brukeren ikke har høy tilgjengelig båndbredde. Innholdet lastes ned før brukeren kan benytte innholdet noe som gir en lang bufring tid. Dersom filene er store, flere GB, så kan dette ta lang tid. Eksempel 1 viste et overslag over hvor stor en fil for en film med en lengde på 2 time ble for bitratene 950 Kbit/sekund og 2000 Kbit/s.

Eksempel 2: Nedlastingstider for ulike filstørrelser.

Filstørrelse for bitrate 950 Kbit/s: 835 MB = 835 * 1024 = 855 040 KB
Filstørrelse for bitrate 2000 Kbit/s: 1758 MB = 1758 * 1024 = 1 800 192 KB

Nedlastingstiden for et ADSL abonnement med nedlastingshastighet på 704 Kbit/s:

Nedlastingstid for filmen med bitrate 950 Kbit/s:
855 040 KB / (704 / 8) Kbit/s = 9716 sekund = 2 timer, 42 minutter.

Nedlastingstid for filmen med bitrate 2000 Kbit/s:
1 800 192 KB / (704 / 8) Kbit/s = 20457 sekund = 5 timer, 41 minutter.

Resultatene er teoretisk nedlastingstid. Det er ikke tatt hensyn til variasjoner i nedlastingshastigheten som for eksempel kommer av ulik trafikk belastning i nettet. I praksis vil nedlastingstiden være noe høyere enn den teoretiske nedlastingstiden..

Eksempelet over viser at å laste ned innhold med den vanligste ADSL forbindelsen per i dag, 704 Kbit/sek, vil ta veldig lang tid noe som er negativt fra brukerensståsted. Brukeren vil ikke få den samme ”on Demand” opplevelsen som streaming av innhold gir. Siden innholdet lastes ned på forhånd er det mulig å kode innholdet med en høyere bitrater enn den som kan brukes for streaming noe som gir bedre videokvalitet enn streaming. Videokvaliteten begrenses ikke av nedlastingshastigheten for dette distribusjonsalternativet. Dersom en velger å distribuere innholdet på en slik måte kan en kode innhold i 2 Mbit/s som vil gi tilnærmet DVD kvalitet. Innholdet lastes ned kun engang noe som er nettverksøkonomisk bra. DRM sørger for at filen er beskyttet og ikke kan misbrukes. En annen fordel er at innholdet kan distribueres brukere i mellom. For

eksempel dersom en bruker har lastet ned innholdet kan han distribuere dette til andre, men disse må da skaffe seg en egen lisens for å benytte innholdet.

6.3.2 Progressiv nedlasting av innhold

Progressiv nedlasting er en hybrid mellom streaming og nedlasting av innhold. Innhold lastes ned og lagres på klient PC'en og en kan spille av innholdet før det er lastet ned i sin helhet. Det betyr at progressiv nedlasting kan benyttes til å dekke inn mangelen av nedlastingshastighet dersom en ønsker å fokusere på videokvalitet. Buffering tiden er lavere enn for vanlig streaming siden avspillingen kan starte før hele filen er lastet ned. Det finnes to scenarioer, det er dersom innholdet er kodet med en lavere bitrate enn nedlastingshastigheten brukeren har tilgang til og dersom innholdet er kodet med en bitrate som er høyere enn nedlastingshastigheten brukeren har tilgang til.

Dersom innholdet er kodet med en bitrate som er lavere enn nedlastingshastigheten brukeren har tilgang til kan avspillingen starte tilnærmet med en gang. Buffering tiden vil i dette tilfellet være tilnærmet som for streaming, mellom 5 - 10 sekunder. Det er mulig fordi innholdet i dette tilfellet lastes ned raskere enn det spilles av på grunn av at nedlastingshastigheten er høyere enn bitraten filen er kodet med. Resten av filen lastes ned mens brukeren spiller av innholdet. Mediaplayeren har funksjonalitet for å finne ut hvor lenge den skal bufre innholdet før den starter avspillingen. En kan forlenge bufringsperioden ved hjelp av Javascript på klientsiden, Mediaplayeren fortsetter å bufre når den står i pause modus. Dersom en ønsker å forlenge bufring tiden kan en sette mediaplayeren i pausemodus i et gitt tidsintervall for deretter å starte avspillingen. Progressiv nedlasting vil i dette tilfellet gjøre distribusjonen av innhold bedre rustet mot variasjoner i nedlastingshastigheten enn tradisjonell streaming. Siden filen lastes ned raske enn det spilles av. Det ble gjort en tilsvarende test som for fast steaming som beviser det, hvor jeg lot filen bufre i fem minutter og deretter belastet aksesslinjen med 40% ekstra trafikk fra andre applikasjoner. Testen ble gjennomført på en 704 Kbit/sek ADSL linje med en fil med bitrate på 650 Kbit/sek.

Det andre scenarioet for progressiv nedlasting av innhold er dersom innholdet er kodet med høyere bitrate enn nedlastingshastigheten brukeren har tilgang til. Mediaplayeren vil da starte en lengre buffering periode før innholdet spilles av. I dette tilfellet vil progressiv nedlasting være mer ømfintlig til variasjoner i nedlastingshastigheten. Variasjonene i nedlastingshastigheten gjør ikke noen spesielle utslag under buffering perioden av filen. Men dersom nedlastingshastigheten synker mye etter at avspillingen av innholdet kan det føre til at bufferet går tomt før hele filen er lastet ned. Mediaplayeren vil da stoppe avspillingen av filen. Det ble gjort samme pratiske test som for forrige scenario.

Det som er et stort problem for progressiv nedlasting i begge scenarioene er brukeropplevelsen dersom avspillingen stopper opp på grunn av at det ikke er lastet ned nok data og brukeren velger å stoppe avpillingen. Når brukeren deretter starter avspillingen på nytt så kan han ikke spole frem til der hvor det stoppet opp første gang. Filen må spilles av fra starten av selv om nesten hele filen er bufret allerede, noe som gir veldig dårlig brukeropplevelse.

For å avdekke funksjonaliteten for dette distribusjonsalternativet måtte det utføres noen tester for å finne ut hvor lang buffering tiden ble for forskjellige alternativer. Testene ble utført på en 704 Kbit/sek ADSL linje. Bitrate'ene som ble valgt for filene var 755 Kbit/sek, 1119 Kbit/sek og 2000 Kbit/sek. De to første ble valgt fordi disse ligger litt over de vanligste ADSL nedlastingshastighetene: 704 Kbit/sek og 1024 Kbit/sek og 2000 Kbit/sek ble valgt på grunnlag av at en ville teste bufferingtiden for et videoklipp med tilnærmet DVD kvalitet. Test resultatene finner en i tabellen under.

Tabell 4: Testresultater fra progressiv nedlasting klipp lengde 26minutter

Bitrate	Kippets lengde	Filstørrelse	Bufferingstid	Teoretisk normal nedlastingstid
755 Kbit/sek	26 min	143 192 KB	2 min 30 sek	27 min 7 sek
1119 Kbit/sek	26 min	209 278 KB	14 min	39 min 38 sek
2000 Kbit /sek	26 min	380 859 KB	47 min	1 t 12 min 7 sek

Tabell 5: Testresultater fra progressiv nedlasting klipp lengde 1time og 40 minutter

Bitrate	Kippets lengde	Filstørrelse	Bufferingstid	Teoretisk normal nedlastingstid
755 Kbit/sek	1 t 40 min	552 978 KB	7 min	1t 44 min 5 sek
1119 Kbit/sek	1 t 40 min	817 493 KB	59 min	2 t 34 min 50 sek
2000 Kbit/sek	1 t 40 min	1 315 071 KB	2 timer 40 min	4 t 9 min 3 sek

Som en kan se av måleresultatene i tabellene over så kan brukeren spille av filen på et mye tidligere tidspunkt med progressiv nedlasting enn normal nedlasting.

En stor ulempe ved progressiv nedlasting er at brukeren ikke kan spole frem eller tilbake i filen før den er ferdig nedlastet. Filen lagres hos brukeren så det er positivt fra aksessleverandørens ståsted, men det er ikke like bra for brukeren i forhold til fast streaming. Når det gjelder videokvalitet så ble det presentert to scenarioer. En kan benytte høye bitrater i kodingen av innholdet men det vil da føre til at buffering tiden blir veldig høy for store filer. For innhold kodet med lavere bitrater kan progressiv nedlasting av innhold fungere som en erstatning til streaming. Dersom en organisasjon ikke har en streamingplattform kan en benytte progressiv nedlasting av innholdet, brukeropplevelsen vil nære tilnærmet lik som streaming bortsett fra spolefunksjonalitets problemene som ble nevnt. Organisasjoner kan

tilby brukere tilnærmet streamingopplevelse uten å investere i en streamingplattform.

Tabell 6: Kort oversikt over fordeler og ulemper ved de ulike distribusjonsmetodene.

Innholds distribusjon med tradisjonell streaming	
Fordeler	**Ulemper**
Brukeren: • Kort buffering tid. • Innholdet tar ikke diskplass. • Spolefunksjonaliteten fungerer.	Brukeren: • Videokvaliteten begrenses av nedlastingshastighet brukeren har tilgang til. Aksessleverandør: • Filen lastes ned fra nettet hver gang den brukes.
Innholds distribusjon med streaming med fast cache funksjonalitet	
Fordeler	**Ulemper**
Brukeren: • Kort buffering tid. • Spolefunksjonaliteten fungerer. • Båndbredden utnyttes mer effektivt enn ved tradisjonell streaming. • Mer robust dersom bitraten i mediafilen ligger tett oppunder nedlastingshastigheten. Aksessleverandør: • Filen lastes ned fra nettet kun en gang. Caches lokalt hos brukeren i en forhåndsdefinert periode.	Brukeren: • Videokvaliteten begrenses av nedlastingshastighet brukeren har tilgang til. • Innhold tar diskplass.
Innholds distribusjon med progressiv nedlasting	
Fordeler	**Ulemper**
Brukeren: • Mulighet for god videokvalitet. • Buffering tiden er redusert i forhold til vanlig nedlasting. Aksessleverandør: • Filen lastes ned fra nettet kun en gang. Lagres lokalt hos brukeren.	Brukeren: • Buffering tiden kan bli veldig lang. • Innholdet tar diskplass. • Spolefunksjonalitet fungerer først etter at hele filen er lastet ned på klient PC'en.

Innholds distribusjon med vanlig fil nedlasting	
Fordeler	**Ulemper**
Brukeren: • Spolefunksjonalitet fungerer. • Mulighet for god videokvalitet. Aksessleverandør: • Filen lastes ned kun en gang. Lagres lokalt hos brukeren.	Brukeren: • Buffering tiden kan bli veldig lang. • Innholdet tar diskplass. • Må selv organisere filen, dvs. slette den når lisensen er utgått osv.

7 Resultater

7.1 Bakgrunn

Hovedpunket i oppgaven var å finne ut hvordan en beskytter innhold med DRM og hvordan en enkelt kan publisere innhold for en slik løsning det vil si krypter og leverer lisenser osv. Det ble brukt DRM programvare fra Microsoft i denne oppgaven, programvaren kan ikke brukes direkte men kun som et fundament i egne løsninger. Som et resultat av oppgaven er skissert to forskjellige oppsett for et VoD system med DRM. Oppsettene fokuserer på en automatisert publiseringsprosess hvor innhold som lastes opp fra innholdsleverandør pakkes, lisensdata lagres og innholdet distribueres til streaming og/eller webservere automatisk.

Oppgaven ser også på hvordan en kan logge aktiviteten på streamingserveren slik at dataene kan benyttes som informasjonskilde for å analysere historiske trender eller brukes som beslutningsgrunnlag når en skal velge ulike kodingsalternativer for innhold. Fordeler og ulemper ved de ulike publiseringsalternativene ble også testet.

7.2 Resultater fra systemoppsettene

Resultatet av systemoppsettene er et system som beskytter rettighetsbelagt innhold på en tilfredstillende måte både for streaming og for nedlasting av innhold. Publiseringen av nytt innhold er enkelt for både innholdsleverandør og aksessleverandør. Ved bruk av DRM kan en beskytte innholdet mye bedre enn ved andre metoder som for eksempel kryptert link systemer eller ulike autentiseringssystemer som kan integreres med streamingserveren. Testresultater fra tester gjort i dette arbeidet viser at bruk av DRM er det eneste alternativet som beskytter innhold bra både for streaming og for nedlasting av innhold. Resultatet av løsningen er at innholdet blir beskyttet tilfredstillende for alle distribusjonsalternativene og åpner med det for en fleksibel løsning for distribusjon av innhold.

En sentral del i oppsettene av prototypsystemene har vært en automatisert publiseringsprosess. Resultatene av prosessen er at løsningen krever minimalt med arbeid fra både innholdsleverandør og aksessleverandør for å publisere nytt innhold. Systemet er enkelt å vedlikeholde siden publiseringsprosessen er automatisert.

Friheten DRM verktøyet fra Microsoft gav resulterte i to forskjellige oppsett av systemet, begge med fokus på en automatisert publiseringsprosess. Alternativene som ble presentert var uploading av innhold med vanlig FTP programvare eller ved

hjelp av en tilpasset uploadingsklient. De to forskjellige systemene har ulike fordeler og ulemper.

Resultatet av systemoppsettet som uploader innholdet med vanlig FTP programvare er et system som gir en effektiv publisering av innholdet. Og er en løsning som er rimelig enkel å implementere og det er enkelt å skalere om løsningen til å bruke flere eller andre streaming og/eller webservere. Pakking og distribusjon av innholdet til streaming og/eller webservere gjøres med Windows servicer. Dette er en enklere implementasjon enn det mer komplekse systemet der en benytter en tilpasset uploadingsprogramvare. Oppsettet er best egnet for scenarioer hvor en ikke trenger å spesifisere mange ulike rettigheter for innholdet som lastes opp. Rettighetene på innholdet som lastes opp er stort sett statisk. Det ble gjennomført tre målinger av to forskjellige filstørrelser for å finne ut hvor effektivt prototypsystemet var. Under testen var pakkeserver og streamingserver plassert på et 100 Mbit LAN. Filene ble kun distribuert til enten webserver eller streamingserver ikke til begge serverne samtidig.

Tabell 7: Gjennomsnittsmåling av pakking av innhold med Windows servicen.

Filstørrelse	Gjennomsnittsmåling pakking av innhold med Windows service	Gjennomsnittsmåling for distribusjon av innholdet til streaming eller webserver
214 300 KB	2 minutter 50 sekunder	2 minutter 35 sekunder
1 346 632 KB	19 minutter 25 sekunder	17 minutter 28 sekunder

Resultatene fra tabell 7 gav en total publiserings tid fra innholdet var lastet opp fra innholdsleverandør til det var distribuert til streamingserver på: 5 minutter 25 sekunder for filen på 214 MB og 36 minutter og 53 sekunder for filen på 1,3 GB.

Publiseringsalternativet ble også testet med et testscript hvor 300 filer med filstørrelser fra 200KB til 12 MB ble lastet opp, pakket og distribuert automatisk. Måling ble gjort fra første fil ble lastet opp til siste fil var distribuert til streamingserver. Tabellen under viser resultatene fra test scriptet og viser effektiviteten i løsningen.

Tabell 8: Måleresultater fra testscript.

Antall filer	Filstørrelse til sammen	Total publiseringstid
300	488 MB	19 minutter 50 sekunder
300	1073 MB	38 minutter 10 sekunder

Uploading med tilpasset programvare vil gi innholdsleverandører større frihet til å sette rettigheter på innholdet som blir publisert. Løsningen krever at innholdsleverandør må installere et klientprogram som brukes til opplastingen av innholdet. I uploadingsklienten settes rettighetene for innholdet som lastes opp og innholdet pakkes av webservicen og distribueres med samme Windows service som i forrige eksempel. Resultatet for dette systemoppsettet er at det passer scenarioer hvor innholdsleverandør ønsker å sette rettighetene for innholdet mer dynamisk etter hvert som innholdet lastes opp til aksessleverandør. Tabellen under viser resultatene fra publiseringsprosessen med tilpasset uploadings klient.

Tabell 9: Måleresultater for publisering med uploadingsklient

Filstørrelse	Gjennomsnittsmåling pakking av innhold med webservice	Gjennomsnittsmåling for distribusjon av innholdet til streaming eller webserver.
91 346 KB	1 minutter 5 sekunder	55 sekunder
214 300 KB	4 minutter 55 sekunder	2 minutter 35 sekunder

Publisering av flere filer samtidig med uploadingsklienten ble testet. Det ble lastet opp fire filer med forskjellig størrelse med en samlet filstørrelse på 303 MB. Resultatene viser at publiseringsprosessen er effektiv også i dette tilfellet.

Tabell 10: Måleresultater for publisering av flere filer med uploadingsklienten.

Antall filer	Filstørrelse til sammen	Total publiseringstid
4	303 MB	5 minutter 35 sekunder

Det ble også laget en plugin for streamingserveren, resultatet var en plugin som logger streaming data til en database. Streamingdataene kan benyttes til ulike formål som for eksempel til å analysere ulike historiske trender for streamingserveren eller som grunnlag for strategiske valg for kodingsalternativer for innholdet.

Systemoppsettene er verifisert ved at det er laget prototypsystemer som beviser at arkitekturen fungerer. Prototyp systemet støtter både opplasting med vanlig FTP programvare og med en tilpasset uploadingsklient.

7.3 Evaluering av distribusjonsalternativer

Det ble testet en del forskjellige distribusjonsalternativer for innhold blant annet ulike former for streaming og nedlastingsalternativer. Testene ble utført i prototyp systemet som støtter alle de ulike alternativene. Testene fokuserte på hvordan de ulike distribusjonsalternativene virker inn på brukervennlighet i løsningen, videokvalitet som er mulig å levere og hva som er mest nettverksøkonomisk sett fra

aksessleverandørens side. Testene hadde som formål å avdekke funksjonaliteten i de ulike løsningene. Mer utfyllende informasjon om de ulike distribusjonsalternativene finner en i kap 6. Under følger en kort beskrivelse av resultatene av publiseringsalternativ testene.

7.3.1 Tradisjonell streaming

Tradisjonell streaming gir en bra brukeropplevelse, brukeren kan spole frem og tilbake i innholdet slik en vil uten begrensinger. Det negative med streaming er at videokvaliteten begrenses av nedlastingshastigheten brukeren har tilgang til. En kan ikke kode innholdet med en høyere bitrate enn nedlastingshastigheten brukeren har tilgang til. Dagens mest utbrette ADSL abonnementer og andre bredbåndsalternativer kan ikke overføre bitrater som er høye nok til å gi en god videokvalitet. For aksessleverandøren er dette en kostbar løsning når det gjelder nettverktrafikk. Hver gang brukeren spiller av filmen må den overføres fra aksessleverandørens server til brukeren.

7.3.2 Fast streaming

Fast streaming bygger på vanlig streaming men har en del tilleggsfunksjonalitet og er en del av Microsoft Media Services 9 Series plattformen.
Tilleggsfunksjonaliteten gjør dette til et bedre alternativ enn tradisjonell streaming både for sluttbruker og for aksessleverandør. Fast streaming gjør det mulig å cache innhold lokalt hos brukeren i en gitt periode. Dersom innhold spilles av flere ganger vil det spilles av lokalt på maskinen dersom det finnes i cachen. Det vil minske nettverkstrafikk noe som er bra for aksessleverandøren. Fast streaming har også en del funksjonalitet som gjør at brukeropplevelsen vil bli bedre enn for tradisjonell streaming. Dersom brukeren har høyere nedlastingshastighet tilgjengelig enn bitraten innholdet er kodet med vil det brukes til å øke cache størrelsen hos brukeren noe som gjør løsningen mer robust mot svingninger i nedlastingshastigheten. Spolefunksjonalitet fungerer også ubegrenset i dette tilfellet. Fast streaming vil ha de samme begrensingene når det gjelder videokvalitet som for tradisjonell streaming.

7.3.3 Normal fil nedlasting av innhold

Nedlasting av innhold setter ingen grenser for bitraten filene kan kodes med. Det vil si at dette alternativet er bra egnet dersom en ønsker å fokusere på videokvalitet i løsningen. Et stor minus er den lange nedlastingstiden som brukeren opplever. Hele filen må i dette tilfellet lastes ned på klient PC'en før avspillingen av innholdet kan startes. Alternativet sparer også nettverkstrafikk siden filen kun lastes ned engang og spilles deretter av lokalt.

7.3.4 Progressiv nedlasting av innhold

Tanken bak bruken av dette publiseringsalternativet var at en kunne dekke inn mangelen av nedlastingshastighet med en lengre buffering tid. Buffering tiden vil være kortene enn for tradisjonell nedlasting av innhold siden nedlasting av innhold kan skje mens innholdet spilles av. Men tester som ble utført i dette arbeidet viser at bufferingtiden blir så lang at det ikke er noe bra alternativ fra brukerens side, se kapittel 6 for flere detaljer. Det er heller ikke mulig å spole i innholdet før hele filen er lastet ned. For eksempel dersom brukeren stanser avspillingen for deretter å starte den vil avspillingen av filen starte fra starten av. Brukeren har ikke mulighet til å spole før hele filen er lastet ned i sin helhet.

7.4 Sammendrag av resultater

Dersom en ønsker å publisere rettighetsbelagt innhold og ønsker å beskytte innholdet mot misbruk og piratkopiereing må DRM brukes til å beskytte innholdet for alle publiseringsalternativene.

Automatisering av publiseringen av innhold kan enkelt implementeres ved hjelp av Windows servicer og eventuelt av en tilpasset uploadningsklient. Testene som er gjennomført gir et tilfredstillende resultat når det gjelder effektiviteten til publiseringsprosessen i løsningen.

Fast streaming vil være best egnet mot variasjon i nedlastingshastigheten siden en da ikke opererer med en fast bufferstørrelse. Bufferet kan da være større enn ved vanlig streaming og gjøre løsningen mer robust i de fleste tilfellene.

Når det gjelder publiseringsalternativene for innhold så er det ikke mulig å levere veldig god videokvalitet med streaming til de fleste bredbåndsabonnenter per i dag. Tallene fra Telenor og NextGenTel sier at en trenger opp mot 2 Mbit/sek for å levere tilnærmet DVD kvalitet på innholdet. Nedlastingshastigheten for de fleste bredbåndsabonnenter er for lav til at det er mulig per i dag. Nedlasting av innholdet kan dekke inn mangelen av båndbredde men dette går på bekostning av buffering tiden som blir meget lang. Tester som ble utført i dette arbeidet viser at progressiv nedlasting ikke er egnet for å dekke inn mangelen på båndbredde på grunn av de negative sidene ved brukervennligheten.

8 Drøfting

8.1 Bakgrunn

Oppgaven startet med å presentere en del teori for VoD systemer som benytter DRM fra Microsoft til å beskytte innhold. Siden DRM produktet fra Microsoft ikke er noe ”stand-alone” system har jeg presentert en arkitektur og laget et lite prototyp system for et VoD system med DRM. Prototypen er laget for å vise at arkitekturen for systemet fungerer. Det blir presentert to forskjellige opplastingsmuligheter for innhold fra innholdsleverandørene til aksessleverandørene. Og jeg har sett på forskjellige distribusjonsalternativer for distribusjon av innhold til brukere. Distribusjonsalternativene jeg har sett nærmere på er ulike former for streaming og nedlasting av innholdet.

8.2 Diskusjon av Microsoft DRM

Fordelene ved å benytte DRM fra Microsoft er at en stor kundemasse har systemer som allerede støtter DRM løsningen, de fleste kjører i dag på Windows plattformen med Mediaplayeren. Microsofts løsning er rimelig, men en ulempe er at DRM fra Microsoft ikke er en komplett løsning. Erfaringer gjort i arbeidet med denne oppgaven tilsier at det kreves en del kompetanse i organisasjoner som ønsker å implementere løsninger med DRM. En annen ulempe er at DRM fra Microsoft kun kan benyttes for filer basert på Microsoft formater: WMV, WMA og ASF. Erfaringen som ble gjort under arbeidet med DRM teknologien er at den er enkel å integrere i nye eller eksisterende systemer. Teknologien er ideell som et fundament for å bygge løsninger som beskytter innholdet med DRM.

Det viste seg også at DRM er den beste metoden for å beskytte innhold. DRM er den eneste metoden som beskytter innhold som distribueres både med streaming og nedlasting. Tradisjonell streaming beskytter i utgangspunket innholdet ”litt” på den måten at innholdet ikke lagres på harddisken til klient PC’en, men behandles kun i minnet på PC’en. Det viste seg også her at en trenger DRM for å beskytte innhold som leveres med streaming. Ved hjelp av ”stream catching” verktøyer kan en ”snappe opp” datastrømmen og lagre den lokalt. På en slik måte kan en piratkopiere streaming innhold som ikke er beskyttet med DRM. WMRecorder er et eksempel på et slikt verktøy som ved å sette opp en lokal proxy kan avlytte datastømmen og lagre en kopi av datastømmen lokalt på maskinen. Datastømmen konverteres senere til en video fil som kan spilles av på PC’en.

Det viste seg også at innhold som streames med fast streaming teknologien trenger DRM beskyttelse siden innholdet caches lokalt hos brukeren. Brukeren trenger ikke

”stream catching” verktøyer for å piratkopiere innhold i dette tilfellet siden innholdet lagres i ”temporary internett files” katalogen og kan kopieres derifra. Det er derfor viktig å benytte DRM i fast streaming løsninger.

Andre metoder for å beskytte innhold er å benytte ulike autentiseringsmoduler eller ”kryptert link” moduler på streamingserveren disse gir heller ikke tilfredstillende beskyttelse av innholdet. Disse alternativene vil begrense tilgangen til innholdet som ligger på streamingserveren til kun brukere med gyldig tilgang, men disse er sårbare mot ”stream catching” verktøyer. Tilgangen til streamen er beskytta men når streamen overføres så overføres den uten noen form for beskyttelse.

Disse erfaringene resulterer i at jeg anbefaler bruk av DRM for alt rettighetsbelagt innhold som skal distribueres over Internett både med streaming og nedlasting av innholdet.

8.3 Diskusjon av arkitektur

8.3.1 Uploading av innhold med FTP programvare

Arkitekturen på uploadserveren kan diskuteres, hva om en har flere hundre innholdsleverandører som alle ønsker tilgang til de to pakkealternativene: normal og batch pakking. Windows servicen kan da ende opp med å måtte sjekke om det er kommet nye filer i flere hundre kataloger hver gang servicen trigges. Dersom tidsintervallet for å trigge servicen er kort vil det ta unødvendig CPU ressurser, det mest hensiktsmessige vil være å benytte ett lenger tidsintervall for eksempel 5 til 10 minutter. En forutsetting for å sette et slik tidsintervall er at publiseringen ikke er tidskritisk det vil si at innholdet ikke må publiseres øyeblikkelig. Dersom publiseringen er tidskritisk bør en operere med kort tidsintervall.

En kan også spørre seg om hvorfor uploadserver/pakkeserver bør kjøres på en egen server. Pakkingen av innholdet er en effektiv prosess, men dersom en skal pakke flere gigabytes med data vil dette ta litt tid og CPU forbruket på serveren vil være svært høyt. Servicen vil også kreve en del diskaksess. Dersom en kjører streamingserveren på samme maskin så kan dette få konsekvenser for streamingserveren. Det kan føre til at streamingserveren ikke klarer å levere nok data til klientene som er tilkoblet. Men det er fullt mulig å kjøre disse komponentene på samme maskin.

8.3.2 Uploading av innhold med en tilpasset applikasjon

Oppgaven beskriver en konseptuelløsning for en tilpasset uploadingapplikasjon og prototypen viser at en slik løsning fungerer. Løsning vil ha både positive og negative sider. Det viste seg at dersom en innholdsleverandør ønsker å definere

rettighetene på innholdet mer dynamisk det vil si at innholdet skal ha mange ulike sammensetninger av rettigheter så vil en slik løsning være å foretrekke.

Det negative er at en slik applikasjon må distribueres og installeres hos de ulike innholdsleverandørene som skal levere innhold til en aksessleverandøren. Det positive er at en slik løsing gir innholdsleverandørene stor fleksibilitet til sette rettigheter, velge pakkealternativ og velge hvordan innholdet skal distribueres til brukerne.

8.4 Måleresultater for innholdsdistribusjon

Måleresultatene for distribusjon av innhold fra pakkeserver til streamingserver som ble presentert i resultatkapittelet vil muligens gi en lengre tid dersom distribusjonen skal skje over Internett siden testene ble utført på et 100 Mbit LAN. Dersom en hadde distribuert filene over Internett ville tiden for distribuering av filene sannsynligvis vært høyere siden overføringshastigheten sannsynligvis hadde vært lavere enn det en hadde over LAN.

8.5 Måleresultater for pakking av innhold

Måleresultatene for pakkingen av innholdet viste at effektiviteten for å pakke innholdet var mye lavere enn det som er oppgitt fra Microsoft. Dokumentasjon viser at effektiviteten av pakkingen av innholdet kan være opp mot 10 MB per sekund. Resultatene jeg fikk lå mye lavere enn dette, testene som ble utført viste at effektiviteten for pakkingen lå på litt over 1 MB per sekund. En av grunnene til dette er at pakkingen av innholdet ble utført på en gammel bærbar maskin med en treg harddisk. (Se tabell 1 oversikt over maskinvare som ble benyttet)

8.6 Lisensserver arkitektur

Den vanligste måten å implementere lisensserver funksjonalitet er ved hjelp av ASP websider (for versjon 7 av DRM SDK'en). En bruker da http requester for å kommunisere mellom lisensserver og webserver. En slik implementering fungerer men det viste seg at dette er en litt foreldet måte å implementere liseserveren siden jeg benytter .NET. Det resulterte i at jeg har valgt å implementere lisensserveren som en webservice. Fordelen er at det da er enkelt å integrere mot lisensserveren ved hjelp et standard grensesnittet for webservices. Integrasjonen mot lisensserveren blir da enklere ved at en benytter standard webservice kall mot lisensserveren. Dersom en kjører webserveren og lisensserveren på to forskjellige maskiner som i prototyp systemet vil et standardisert grensesnitt være en fordel fordi det er enkelt å integrere de to maskinene.

8.7 Streamingserver plugin

Testing av logging pluginen og analyse av loggdataene som pluginen leverte viste at det er vanskelig å følge sesjoner som en klient har mot en streamingserver. Det blir generert logdata og en ny sesjon mot serveren for hver gang en bruker for eksempel spoler frem eller tilbake i filen. Det viser seg at logdataene derfor ikke kan benyttes til å følge klient sesjoner på streamingserveren. Men at loggene fra streamingserveren kan benyttes for å hente ut data om historiske trender eller belastninger på streamingserveren.

8.8 Videre arbeid

Det er vist at arkitekturen for VoD systemet fungerer gjennom implementeringen av prototypen. Prototypen av systemet er på ingen måte ferdig utviklet, den inneholder kjerne komponentene men det er fortsatt områder som kan videreutvikles. Dette avsnittet vil presentere de områdene hvor jeg mener det trengs videreutvikling.

Prototypen benytter Windows Media Digital Rights Management SDK'en versjon 7. Etter at dette arbeidet ble påbegynt er det kommet ut en nyere versjon av denne SDK'en. Den siste versjonen er versjon 9. API'et i denne SDK'en er mye likt API'et for versjon 7, men det er kommet noe ny funksjonaltet i versjon 9. Jeg vil spesielt trekke frem LiveDRM, LiveDRM gjøre det enkelt å benytte DRM i Windows Media Encoder'en. En kan da pakke innholdet med DRM i samme prosess som en koder innholdet og benytte DRM for live broadcasts av innhold.

DRM løsningen bør ha funksjonalitet for å logge antall lisenser en bruker mottar. Dette for å begrense mulighetene for svindel i løsningen. Dersom en bruker for eksempel skaffer seg en lisens til innholdet og flere brukere har kjennskap til brukernavn og passord for kontoen kan en skaffe seg lisenser til innholdet på flere maskiner. Løsningen bør derfor ha en antall begrensning for antall lisenser som leveres til en bruker for et og samme innhold.

Det som i fremtiden kan gjøre Video on Demand distribusjon over bredbånd meget interessant er dersom en utvikler løsninger for å bestille variabel båndbredde. Brukere kan da bestille en høyere båndbredde i en gitt periode for eksempel i to timer mens en streamer en film. Høyere båndbredde vil gi brukere høyere nedlastingshastighet noe som fører til at en kan streame innhold som er kodet med høy bitrate. Det vil gi bedre videokvalitet for innholdet enn det som er mulig å levere med streaming over dagens mest populære bredbåndsabonnementer. NextGetTel gjennomførte helgen 9. – 11. mai 2003 en vellykket fullskala test hvor

de testet bruk av variabel båndbredde. Det var en test hvor de skrudde opp kapasiteten i hele nettet, men løsningen som er skissert her går ut på å øke nedlastingshastigheten for enkelte brukere i en gitt periode. Dette er en mer kompleks løsning som blant annet NextGenTel ser på mulighetene for å implementere.

9 Konklusjon

I oppgaven har jeg sett på hvordan en kan sette sammen et Video on Demand system som beskytter rettighetsbelagt innhold med Digital Rights Management mot blant annet piratkopiering og ubegrenset bruk av innholdet. Innholdet krypteres og tilgangen til innholdet kan effektivt administreres med lisenser. Lisensene inneholder en nøkkel for å dekryptere innholdet og informasjon om rettighetene brukerne har til innholdet som dekkes av lisensen.

Digital Rights Management er egnet til å beskytte innhold mot piratkopiering og annen form for misbruk av rettighetsbelagt materiale. På grunnlag av tester som ble utført i oppgaven kan en konkludere med at DRM må benyttes for å beskytte innholdet både for streaming og nedlasting. DRM plattformen fra Microsoft gir stor fleksibilitet og er en solid plattform som en kan bygge inn i nye eller eksisterende systemer. DRM beskytter innholdet både ved streaming og ved nedlasting av innholdet, noe som gir god fleksibilitet til å velge ulike distribusjonsalternativer for innholdet.

Arkitekturen og designet som er beskrevet i oppgaven gjør publiseringen av innhold enkel ved at den automatiserer publiseringen av innhold. En slik automatisering av publiseringsprosessen vil minimere arbeidet hos innholdsleverandør og aksessleverandør for å publisere innhold i løsningen. Det ble presentert to ulike alternativer for publiseringen av innholdet: uploading med vanlig FTP programvare og uploading med en tilpasset uploadingsklient. Begge alternativene automatiserer publiseringen av innholdet, men en tilpasset uploadingsklient vil gi innholdsleverandører større fleksibilitet når det gjelder å sette ulike rettigheter, pakkingsalternativer og publiseringsmetoder på innholdet som lastes opp. De ulike systemoppsettene er testet og verifisert ved at det er laget prototyp systemer som tester oppsettene, begge oppsettene fungerte bra.

Distribusjonsalternativet som er best egnet for VoD med DRM er fast streaming. Dette er det beste distribusjonsalternativet både for brukere og aksessleverandører. Det er den løsningen som gir best brukeropplevelse for brukeren. Fast streaming løsningen har en del tilleggsfunksjonalitet som tradisjonell streaming ikke har og som gjør dette distribusjonsalternativet til det beste. Løsningen er i de fleste tilfeller bedre rustet mot variasjoner i nedlastingshastighet siden en ikke opererer med en fast bufferstørrelse som for vanlig streaming. Siden innholdet caches lokalt i en gitt periode er dette ”nettverksøkonomisk” bra sett fra aksessleverandørens side. Det som har veid tungt når jeg mener at fast streaming er det beste alternativet er ”on Demand” aspektet ved løsningen og at innholdet caches hos brukeren. Produktet

som selges skal leveres øyeblikkelig og ikke etter mange timer som ville vært tilfellet ved nedlasting av innholdet.

Manglende nedlastingshastighet kan dekkes inn med progressiv nedlasting av innhold dersom en ønsker å fokusere på videokvalitet. Tester gjennomført viser at progressiv nedlasting kan brukes dersom en ønsker å fokusere på videokvalitet, men brukervennligheten for løsingen gjør at dette ikke er det beste distribusjonsalternativet. Brukeren kan ikke spole i innholdet før hele filen er lastet ned og buffering tiden er lang. Progressiv nedlasting er best egnet som en tilleggstjeneste til fast streaming, slik at brukeren selv kan velge distribusjonsalternativ.

10 Forkortelser

VOD	Video On Demand
DRM	Digital Rights Management
SAP	Secure Audio Path
COM	Common Object Model
SDK	Software Development Kit
API	Application Programming Interface
ASDL	Asymmetric Digital Subscriber Line
IP	Internet Protocol
HTTP/1.0	Hypertext Transfer Protocol versjon 1.0
HTTP/1.1	Hypertext Transfer Protocol versjon 1.1
FTP	File Transfer Protocol
RTSP	Real Time Streaming Protocol.
MMS	Microsoft Media Server protocol.
SOAP	Simple Object Access Protocol
QoS	Quality of Service
WMV	Windows Media Video
WMA	Windows Media Audio
ASF	Advanced Systems Format
RSA	Krypterings/digitalsignatur algoritme oppfunnet av Rivest, Shamir og Adelman i 1976
ASP	Active Server Pages

11 Referanser:

[1] Microsoft
http://www.microsoft.com
Dato: 1. februar 2003

[2] Microsoft Digital Rights Management
http://www.microsoft.com/windows/windowsmedia/drm.aspx
Dato: 1 februar 2003

[3] RealNetworks – Helix DRM
http://www.realnetworks.com
Dato: 1.mars 2003

[4] DMDSecure
http://www.dmdsecure.com/
Dato: 1.mars 2003

[5] Adobe Norge
http://www.adobe.no
Dato 1. februar 2003

[6] Windows Media
http://www.microsoft.com/windows/windowsmedia/
Dato: 1 februar 2003

[7] Microsoft Developer Network
http://msdn.microsoft.com
Dato 1. februar 2003

[8] Dokumentasjon – DRM SDK'en
Windows Media Rights Management SDK
SDK fra Microsoft

[9] Internet Engineering Task Force
http://www.ieft.org
Dato: 1 februar 2003

[10] RFC 1945, Hypertext Transfer Protocol - HTTP/1.0.

[11] RFC 2068, Hypertext Transfer Protocol - HTTP/1.1.

[12] RFC 2326, Real Time Streaming protocol - RTSP

[13] Microsoft Media Server protocol - MMS
http://msdn.microsoft.com/
Dato: 1 februar 2003

[14] RFC 959, File Transfer Protocol (FTP).

[15] Whitepaper: Windows Media 9 Series Deployment Guide
Microsoft New Media Platforms Division
Publisert: desember 2002

[16] Whitepaper: Logging Model for Windows Media Services 9 Series
av Huseyin Koyun
New Media Platform Division, Microsoft
Publisert: mars 2003

[17] API referanse for plugins utvikling for streamingserveren
http://msdn.microsoft.com/library/en-us/wmsrvsdk/htm/customvbobjects.asp
Dato: 31. mars 2003

[18] SF Anytime
http://www.sf-anytime.com/
Dato: 1. februar 2003

Vedlegg A – kildekode fra prototyp

CD med de ulike delene av prototypen.

- DRM rammeverket – DRM rammeverket beskrevet i oppgaven.
- Pakkeservicen – Windows service som pakker innhold.
- Distribusjonsservice for intern nett. – Windows service som kopierer innhold pakkeservice til streaming og/eller webserver.
- Distribusjonsservice for Internett (FTP) – Windows service som distribuerer innhold til streaming og/eller webserver med FTP.
- Lisensserver – Webservice som leverer lisenser.
- Web applikasjon – Demonstrasjons webapplikasjon.
- SQL scripts til databasen – diverse scripts for å opprette database, stored procedures og tabeller.
- Tilpasset uploading klient – Windows Applikasjon for å laste opp innhold til aksessleverandør.
- Packaging WebService – Webservice som pakker innhold som lastes opp fra innholdsleverandør.
- Log plugin – plugin til streamingserveren for å behandle logdata.

CD plasseres i lomme HER!!!

Vedlegg B - meldingsformat

Formatet på meldingene fra streamingserveren ser slik ut:

```
<XML>
<Summary>
169.254.9.76 2003-03-30 21:25:58 - /EppaTV/pearlharbor_750k.wmv 216 7 4 200 {3300AD50-2C39-46c0-AE0A-ED74D85A7D4D} 9.0.0.2980 en-GB WMFSDK/9.0.0.2980_WMPlayer/9.0.0.2980 - wmplayer.exe 9.0.0.2980 Windows_2000 5.0.0.2195 Pentium 969 91346223 3519145 rtsp TCP - - - 3054572 3054572 382 382 0 0 0 0 0 0 1 1 100 169.254.72.23 streamingserver 1 0 - 2 file://C:\WMPub\WMRoot\movies\pearlharbor_750k.wmv mms://169.254.72.23/EppaTV/pearlharbor_750k.wmv pearlharbor_750k.wmv - - 0
</Summary>
<c-ip>0.0.0.0</c-ip>
<date>2003-03-30</date>
<time>21:25:31</time>
<c-dns>-</c-dns>
<cs-uri-stem>rtsp://169.254.72.23/EppaTV/pearlharbor_750k.wmv</cs-uri-stem>
<c-starttime>216</c-starttime>
<x-duration>7</x-duration>
<c-rate>4</c-rate>
<c-status>200</c-status>
<c-playerid>{3300AD50-2C39-46c0-AE0A-ED74D85A7D4D}</c-playerid>
<c-playerversion>9.0.0.2980</c-playerversion>
<c-playerlanguage>en-GB</c-playerlanguage>
<cs-User-Agent>WMFSDK/9.0.0.2980_WMPlayer/9.0.0.2980</cs-User-Agent>
<cs-Referer>-</cs-Referer>
<c-hostexe>wmplayer.exe</c-hostexe>
<c-hostexever>9.0.0.2980</c-hostexever>
<c-os>Windows_2000</c-os>
<c-osversion>5.0.0.2195</c-osversion>
<c-cpu>Pentium</c-cpu>
<filelength>969</filelength>
<filesize>91346223</filesize>
<avgbandwidth>3519145</avgbandwidth>
<protocol>rtsp</protocol>
<transport>TCP</transport>
<audiocodec>-</audiocodec>
<videocodec>-</videocodec>
<c-channelURL>-</c-channelURL>
<sc-bytes>-</sc-bytes>
<c-bytes>3054572</c-bytes>
<s-pkts-sent>-</s-pkts-sent>
<c-pkts-received>382</c-pkts-received>
<c-pkts-lost-client>0</c-pkts-lost-client>
<c-pkts-lost-net>0</c-pkts-lost-net>
<c-pkts-lost-cont-net>0</c-pkts-lost-cont-net>
<c-resendreqs>0</c-resendreqs>
<c-pkts-recovered-ECC>0</c-pkts-recovered-ECC>
<c-pkts-recovered-resent>0</c-pkts-recovered-resent>
<c-buffercount>1</c-buffercount>
<c-totalbuffertime>1</c-totalbuffertime>
<c-quality>100</c-quality>
<s-ip>-</s-ip>
<s-dns>-</s-dns>
<s-totalclients>-</s-totalclients>
<s-cpu-util>-</s-cpu-util>
<cs-url>mms://169.254.72.23/EppaTV/pearlharbor_750k.wmv</cs-url>
<cs-media-name>pearlharbor_750k.wmv</cs-media-name>
<cs-media-role>-</cs-media-role>
</XML>
```

Vedlegg C – måleresultater

Tabellen viser resultatene for målingene av pakking av innhold med en Windows Service.

Filstørrelse	Gjennomsnittsmåling
214 300 KB	3 minutter 01sekunder
214 300 KB	2 minutter 50 sekunder
214 300 KB	2 minutter 39 sekunder
1 346 632 KB	19 minutter 45 sekunder
1 346 632 KB	18 minutter 10 sekund
1 346 632 KB	20 minutter 20 sekund

Tabellen viser måleresultater for distribuering av innhold med FTP til streaming eller webserver med en Windows service.

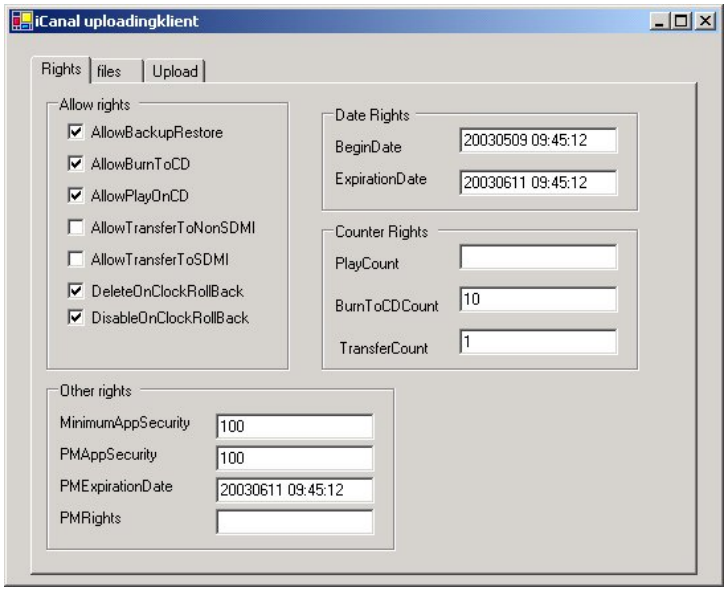
Filstørrelse	Gjennomsnittsmåling
214 300 KB	2 minutter 55 sekunder
214 300 KB	2 minutter 20 sekunder
214 300 KB	2 minutter 25 sekunder
1 346 632 KB	17 minutter 0 sekunder
1 346 632 KB	18 minutter 10 sekund
1 346 632 KB	17 minutter 15 sekund

Vedlegg D - skjermbilder fra prototypen

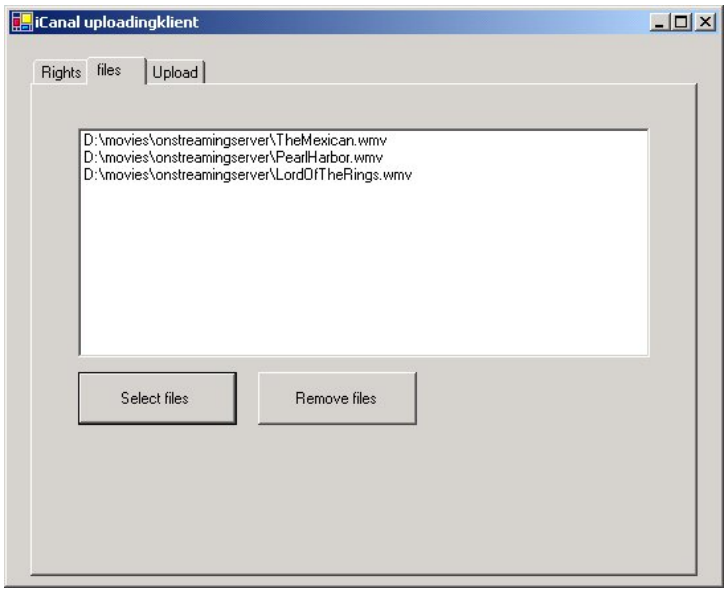
Uploadingsklient

Skjembildene under viser et forslag til hvordan en tilpasset uploadingsklient kan se ut. De er hentet fra uploadingsklienten som ble laget som en del av prototypen. Innholdsleverandør må gjennom tre steg når han laster opp innholdet:

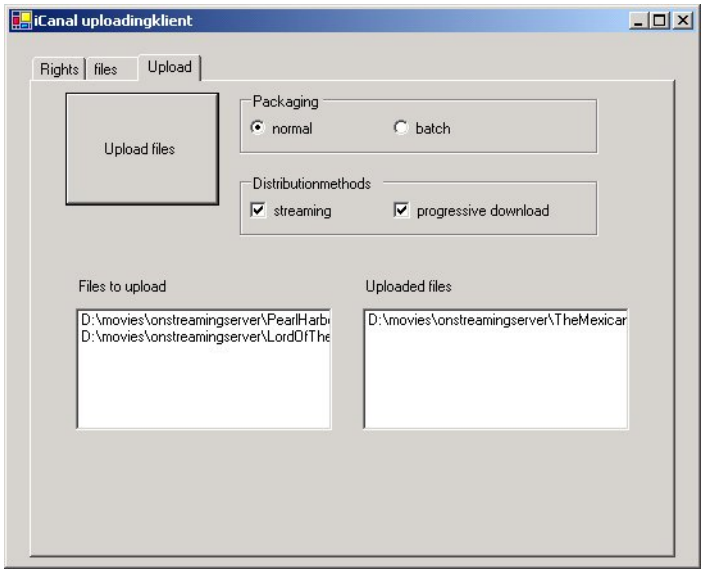
- Steg 1: Innholdsleverandør setter rettighetene på innholdet som lastes opp.
- Steg 2: Innholdsleverandør velger filene som skal lastes opp til aksessleverandøren.
- Steg 3: Innholdsleverandør velger hvordan filene som lastes opp skal pakkes (normalt eller batch) og om de skal gjøres tilgjengelig for brukeren som streaming og/eller nedlasting.



Figur 23: Steg 1 for uploading av innhold.



Figur 24: Steg 2 for uploading av innhold.



Figur 25: Steg 3 for uploading av innhold.

Webdemonstrasjonssider

Under følger en skjermbildeserie fra web delen av prototypen. Den ble laget for å demonstrere hvordan de ulike lisensleveringsalternativene som er blitt beskrevet i oppgaven vil fungere for brukeren. Demonstrasjonssidene er rettet mot min oppdragsgiver, Telenor iCanal, men prinsippene som brukes er generelle og kan brukes for alle VoD løsninger med DRM basert på Microsoft teknologi.

Skjermbildet under viser en produktkatalog som presenteres for brukeren. Brukeren velger produktet han ønsker å kjøpe.

Figur 26: Skjermbilde produktkatalog

Brukeren tas til en detaljside for produktet, dersom brukeren ønsker å kjøpe produktet velger han kjøp og en betalingsdialog startes.

Figur 27: Skjermbilde detaljside

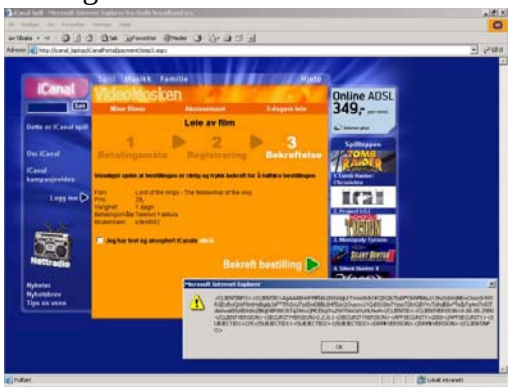
Første steg i betalingsdialogen startes.

Figur 28: Skjermbilde kjøpsdialog steg 1

I det andre steget i betalingsdialogen autentiserer brukeren seg.

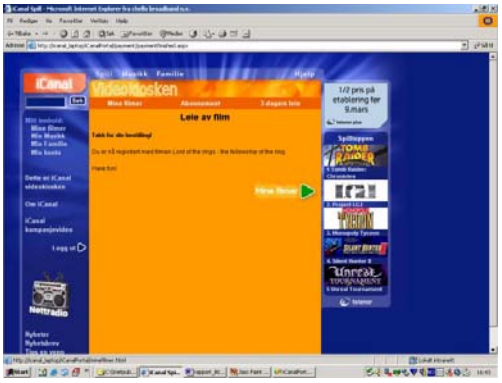
Figur 29: Skjermbilde kjøpsdialog steg 2

I steg tre i betalingsdialogen kommer DRM funksjonaliteten inn. Skjermbildet under viser at det hentes informasjon om DRM programvaren hos klienten. Popupen er med for å vise informasjonen som hentes inn fra klienten. Den vil ikke være med i en implementasjon av løsningen.



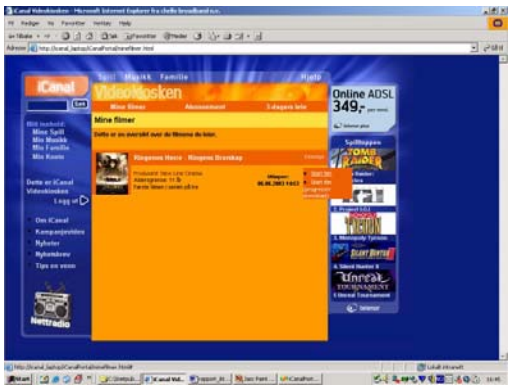
Figur 30: Skjermbilde kjøpsdialog steg 3

Kjøpet av produktet bekreftes og ved hjelp av informasjonen om klienten som ble hentet inn i steg tre av betalingsdialogen og keyID'en til produktet genereres og lagres en lisens til produktet.



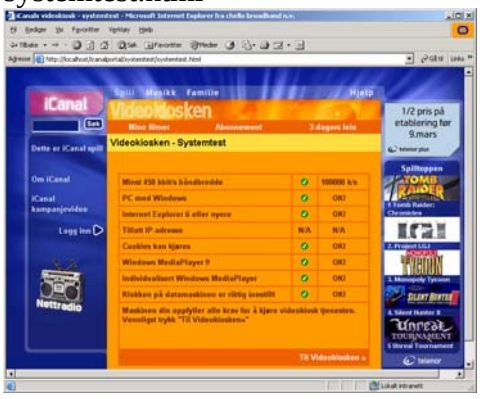
Figur 31: Skjermbilde kjøpsdialog bekreftelse

Lisens til produktet er allerede lagret på klient PC'en. Systemet er klart til avspilling av innholdet.



Figur 32: Skjermbilde mine filmer

Figuren viser siden hvor systemtesten av klient Pcen gjøres. Kildekode finnes i Icanalportalt\systemtest\ systemtest.html



Figur 33: Systemtest av klient PC.