Tomi Lindroos

Playout-järjestelmän arkistointi

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikka

Insinöörityö

17.5.2017

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Tomi Lindroos Playout-järjestelmän arkistointi 20 sivua + 3 liitettä (8 sivua) 17.5.2017
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	Yliopettaja Auvo Häkkinen Playout Manager Teemu Pietilä

Insinöörityön tarkoitus oli suunnitella ja toteuttaa sovellus, joka siirtää videomateriaalia Windows 2012 R2 -palvelimelta DAC ALTO -palvelimelle. Sovellus on osa playout-järjestelmää, jossa arkistoidaan videomateriaali, jota ei enää lähitulevaisuudessa käytetä. Sovellus siirtää tiedostoja määriteltyjen aikajaksojen väliltä automaattisesti sekä manuaalisesti käyttäjän määrittelemien päivien väliltä. Sovellukseen toteutettiin käyttöliittymä, josta ilmenee siirtymässä olevat tiedostot, arvioitu tiedostonsiirtonopeus ja tiedostonsiirron edistyminen. Sovellus ohjelmoitiin pääsääntöisesti C#-kielellä, ja sen käyttöliittymä pohjautui Windows Forms -kirjastoon.

Insinöörityössä tarkasteltiin sovelluksen toteutuksen lisäksi myös arkistoinnin kannalta oleellisten komponenttien vuorovaikutusta työn tilaajan konkreettisessa playout-järjestelmässä. Insinöörityössä käytiin läpi playout-järjestelmässä liikkuvan videomateriaalin elinkaari vaiheittain videomateriaalin toimituksesta arkistointiin asti sekä se, kuinka sen eri vaiheet on toteutettu.

Insinöörityön päätteeksi todettiin, että toteutettu sovellus oli toimiva ja se täytti työn tilaajan asettamat tekniliset määritelmät. Sovellusta jatkokehitetään työn tilaajan tarpeiden mukaan.

Avainsanat	Windows Server 2012 R2, arkistointi, playout

Author(s) Title Number of Pages Date	Tomi Lindroos Archiving playout system 20 pages + 3 appendices (8 pages) 17 May 2017
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Auvo Häkkinen, Principal Lecturer Teemu Pietilä, Playout Manager

The purpose of this thesis work was to design and implement an application that transfers video material from Windows 2012 R2 -server to DAC ALTO -server. The application is a part of a playout system, where its purpose is to archive video material no longer needed in the near future. The application archives the video files between set dates automatically and manually by the end user. A graphical user interface was implemented to the application that visualizes the estimated transfer speed and progression of the ongoing transaction and the files to be moved. The application was written in C# and its GUI as a Windows Forms application.

In addition to going through the implementation of the application, the study also observed the interactions between the components relevant to the archiving process in the playout system. The study observed the life cycle of the video material that went through the playout system and how each step had been implemented.

The outcome of the study turned out to be a working solution and it met the client's technical requirements. The application will be further developed when the client needs more functionalities or changes in it.

Keywords	Windows Server 2012 R2, archiving, playout

Sisällys

Lyhenteet

ALTO	DAC-yrityksen arkistointiin suunniteltu palvelin.
BEPlay	EVS-palvelimen apuohjain.
crystal	Lawo-yrityksen miksauspöytä.
DAC	Disk Archive Company. Yritys joka valmistaa massamuistilaite ratkaisuja. ALTO-palvelimen valmistaja.
Evertz	Videomatriisi, jolla reititetään päätelaitteisiin tulevaa videosignaalia playout-järjestelmän laitteille.
EVS-XS2	EVS-yrityksen videopalvelin.
FTP	File Transfer Protocol. Tiedostonsiirtoprotokolla.
IPDirector	EVS-palvelimien graafinen käyttöliittymä.
MAID	Massive array of idle drives. Tekniikka jolla, pyritään parantamaan kiintolevyjen hintaa, kulutusta sekä jäähtyvyyttä suuremmalla arsenaalilla kovalevyjä. Tekniikka perustuu siihen, että ainoastaan tarvittavat kiintolevyt ovat päällä.
MCR	Master Control Room. Tila jossa playout-järjestelmää hallinnoidaan.
MTG	Modern Times Group MTG AB. Ruotsalainen mediayhtiö, joka omistaa Viasatin.
MXF	Material Exchange Format. Ammattikäytössä oleva videotiedostoformaatti, joka tukee aikakoodeja sekä metatietoja.
NAS	Network-attached storage. Verkossa oleva palvelin, joka toimii tallennusjärjestelmänä.

NEP	NEP Finland Oy. Insinöörityön tilaaja, jolle Viasatin Urheilu-, Jalkapallo ja Jääkiekko -kanavien toteutus on ulkoistettu.
OASYS	BroadStream-yrityksen järjestelmä, joka on työn tilaajan playout-järjestelmän pääkomponentti.
Playout	Termi, jolla tarkoitetaan TV/Radio-tuotannon välittämistä ohjelmatoiminnan harjoittajalta lähetysverkkoon.
RAID	Redundant array of independent disks. Tekniikka, jolla pyritään parantamaan kiintolevyjen vikasietoisuutta ja nopeutta yhdistämällä useita levyjä yhdeksi loogiseksi levyksi.
SATA	Serial ATA. Liitin massamuistilaitteen kiinnittämiseksi tietokoneeseen.
SDI	Serial digital interface. Standardi digitaalisen videon siirtämiseen koaksiaalikaapeleilla.
Viasat	Oy Viasat Finland AB. Insinöörityön tilaajan asiakas, joka toimittaa satelliitti-TV-kanavia.
V_pro8	Lawo-yrityksen videoprosessori.

1 Johdanto

Insinöörityön tavoitteena oli suunnitella ja toteuttaa sovellus, joka siirtää videomateriaalia Windows 2012 R2 -palvelimelta DAC ALTO -palvelimelle. Sovellus on osa playout-järjestelmää, jossa arkistoidaan videomateriaali, jota ei enää lähitulevaisuudessa käytetä.

Sovellus siirtää tiedostoja määriteltyjen aikajaksojen väliltä automaattisesti sekä manuaalisesti käyttäjän määrittelemien päivien väliltä. Sovellukseen toteutettiin käyttöliittymä, josta ilmenevät siirtymässä olevat tiedostot, arvioitu tiedostonsiirtonopeus ja tiedostonsiirron edistyminen. Sovellus ohjelmoitiin pääsääntöisesti C#-kielellä, ja sen käyttöliittymä pohjautui Windows Forms -kirjastoon. Työ tehtiin NEP Finland Oy:lle keväällä 2017.

Insinöörityössä tarkasteltiin sovelluksen toteutuksen lisäksi myös arkistoinnin kannalta oleellisten komponenttien vuorovaikutusta työn tilaajan konkreettisessa playout-järjestelmässä. Insinöörityössä käytiin läpi playout-järjestelmässä liikkuvan videomateriaalin elinkaari vaiheittain videomateriaalin toimituksesta arkistointiin asti sekä se, kuinka sen eri vaiheet on toteutettu.

Työn tilaajan toiveesta insinöörityössä tehdyn sovelluksen sekä DAC ALTO -palvelimen vuorovaikutusta ja toteutusta ei käsitellä julkisessa versioissa insinöörityössä salassapitosopimuksen vuoksi. Julkisesta versiosta puuttuvia osioita havainnollistetaan sovelluksen sekä perinteisen FTP-palvelimen välisenä vuorovaikutuksena.

2 Playout-järjestelmä

Playout on termi, jolla tarkoitetaan TV/Radio-tuotannon välittämistä ohjelmatoiminnan harjoittajalta lähetysverkkoon. Playout-järjestelmä on kokoonpano, jolla tämä toteutetaan. Playout-järjestelmiin on useita ratkaisuja, ja ne voivat koostua useista eri komponenteista. Tässä insinöörityössä keskitytään pääasiassa konkreettiseen järjestelmään, joka löytyy työn tilaajan, NEP Finland Oy:n, tiloista.

NEP Finland Oy toimittaa Oy Viasat Finland AB:n -tuoteperheen TV-kanavia: Urheilu, Jalkapallo sekä Jääkiekko. Playout-järjestelmää hallinnoidaan NEP:n lähetysyksiköstä, eli MCR-tiloista (engl. Master Control Room).

2.1 Toimintaperiaate

Viasat tilaa asiakkailtaan toivomiansa live-lähetyksiä sekä makasiiniohjelmia. Videomateriaali toimitetaan joko suoraan asiakkaan kanssa järjestettyjen valokuitukaapeleiden välityksellä tai satelliittien välityksellä YLE:n (Yleisradio Oy) kytkentäkeskukseen. Kytkentäkeskuksesta sekä asiakkaiden valokuitukaapeleiden välityksellä tuleva videosignaali on matrisoitu NEP:n playout-järjestelmään.

Valokuitukaapelit matrisoidaan V_pro8-videoprosessoriin, jolla synkronisoidaan kuva sekä summataan äänet. Tämä matrisoidaan edelleen live-lähetyksiä varten OASYS-järjestelmään sekä EVS-palvelimille. OASYS-järjestelmä huolehtii ohjelmalistan järjestyksessä ajamisesta lähetysvirtaan. Videomateriaalin halutut osiot leikataan ohjelmistoon sekä arkistointia varten EVS XS -palvelimella. Ohjelman leikkaamisen jälkeen videomateriaali siirretään lähiverkossa olevalle NAS-palvelimelle (engl. network-attached storage). Ohjelmat pysyvät NAS-palvelimella määritellyn ajan, josta videomateriaali on suoraan käytettävissä OASYS-järjestelmässä. Tämän jälkeen ohjelmisto siirretään arkistoitavaksi DAC ALTO -palvelimelle, josta videomateriaali voidaan tarvittaessa palauttaa.

NEP:n playout-järjestelmään kuuluu myös useita muita komponentteja, kuten signaalimuuntimia, streamer-lähettimiä sekä useita muita toiminnallisuuksia edellä esitellyissä komponenteissa. Kyseiset toiminnallisuudet eivät kuitenkaan vaikuta oleellisesti playout-järjestelmän arkistointiin.

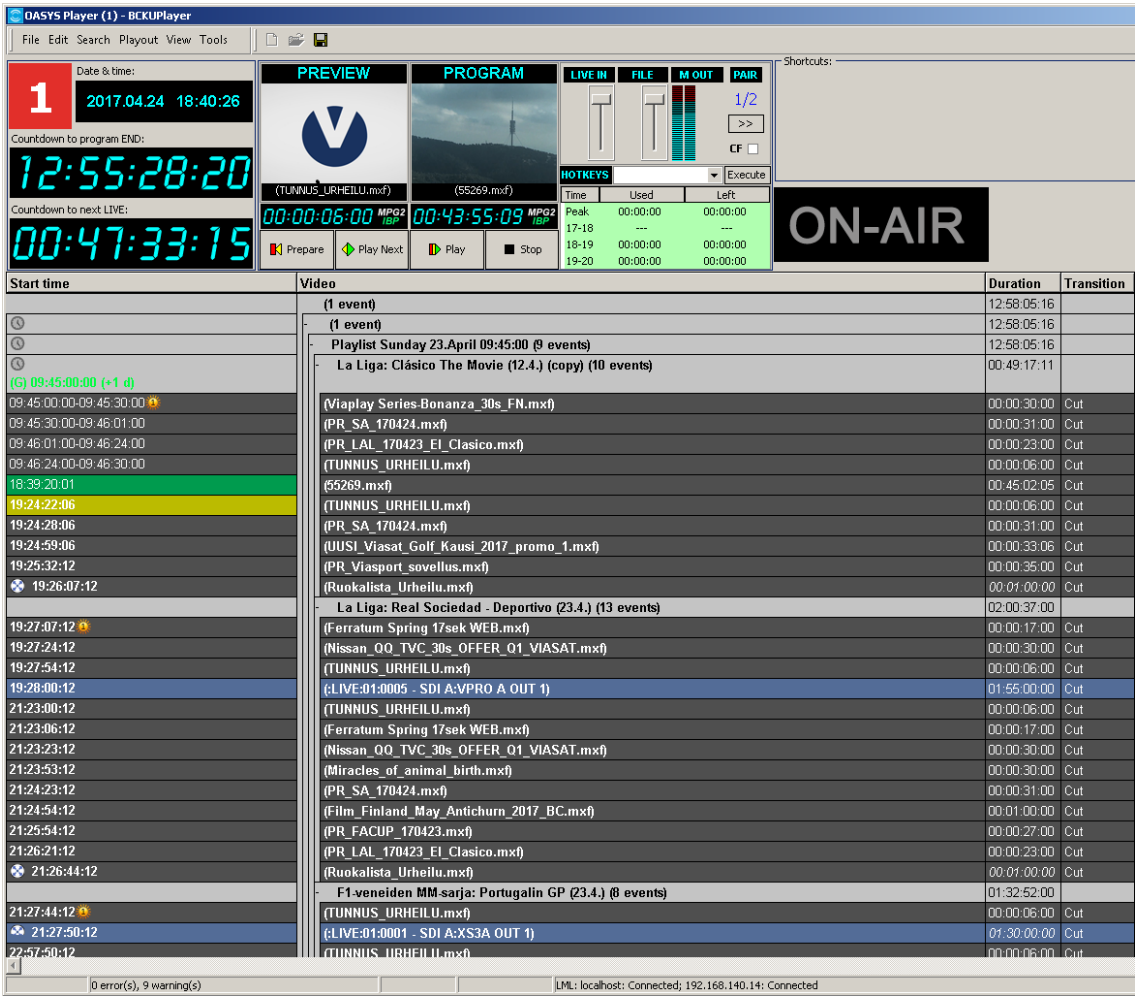
2.2 OASYS BroadStream -järjestelmä

BroadStream-yrityksen OASYS-järjestelmä on NEP:n playout-järjestelmän pääkomponentti, jonka ympärille playout-järjestelmä on rakennettu. OASYS-järjestelmä on ratkaisu, joka sisältää yleisimmät playout-palvelut, kuten grafiikan hallinnan, materiaalin hallinnan ja toiston (engl. channel in a box). [1.] OASYS-järjestelmä räätälöidään lähtökoh-

taisesti asiakkaan tarpeiden mukaan, joten insinöörityössä keskitytään tarkastelemaan NEP:n konfiguraatiota.

OASYS-järjestelmä vastaa videomateriaalin ulosajamisesta sekä ajastamisesta lähetysvirtaan. Jokaista TV-kanavaa kohti on yksi Player-palvelin, joka toistaa soittolistalta siihen määriteltyä videomateriaalia. Player-palvelin toistaa MXF-tiedostoformaatissa (engl. Material Exchange Format) olevaa videomateriaalia NAS-palvelimelta tai suoraa videosignaalia. Player-palvelin tukee myös muita tiedostoformaatteja, mutta toimintaperiaatteiden kannalta on standardoitu MXF-tiedostoformaatin pääsääntöinen käyttö. Järjestelmä on lähiverkon ylitse yhteydessä NAS-palvelimeen, josta se hakee toistettavan videomateriaalin. Suoraa signaalia voidaan reitittää V_pro8-videoprosessorin ulosläh-tevästä videosignaalista tai EVS XT2 -palvelimen ohjelmakanavista. Player-palvelimen uloslähtevä signaali on matrisoitu TX-kanavaan, josta se lähtee Ruotsin Modern Times Group MTG AB:lle sekä YLE:n kytkentäkeskuksen kautta Digita Oy:n verkonhallinta-keskukseen.

Player-palvelimia ohjataan kuvassa 1 esitellyllä käyttöliittymällä.



Kuva 1. OASYS-järjestelmän Player-palvelimen käyttöliittymä.

Kuvan harmaalla taustalla olevat videomateriaali on haettu NAS-palvelimelta. Ylempi sinisellä taustalla oleva tapahtuma (engl. Event) toistaa suoraa V_pro8-videoprosessorin VPRO-A1 -uloslähdön videosignaalia. Alempi sinisellä taustalla oleva tapahtuma toistaa videosignaalia EVS XT2 -palvelimen ohjelmakanavalta.

3 Videomateriaalin elinkaari

Tässä osioissa tarkastellaan tarkemmin arkistoinnin kannalta oleellisia komponentteja sekä niiden vuorovaikutusta aloittaen videomateriaalin toimituksesta aina arkistointiin asti.

3.1 Signaalitiet

Toimittaja toimittaa videomateriaalia joko SDI-signaalina tai datana NEP:n laitehuoneen päätelaitteisiin. Eri toimittajilla on eri päätelaitteet riippuen sopimuksesta sekä toimitettavan videomateriaalin formaatista. SDI-signaalina tulevat toimitukset tulevat lähtökohtaisesti YLE:n kytkentäkeskuksen kautta päätelaitteisiin. Päätelaitteet matrisoidaan Evertz-videomatriisiin, josta niitä voidaan reitittää rajattomasti useamman laitteen sisääntuloon. Evertz-videomatriisilla voidaan esimerkiksi reitittää sisääntuleva videosignaali V_pro8-videoprosessoriin, jolla summataan suomenkieliset selostukset ja tämä uudelleen matrisoidaan takaisin YLE:n kytkentäkeskukseen lähetystä varten.

3.2 V_pro8-videoprosessori

V_pro8 on Lawo yrityksen digitaalinen 8-kanavainen videoprosessori, jota käytetään videokanavien sekä ääniraitojen yhdistelemiseen. Se sisältää reitittimen, jolla voi vapaasti reitittää 8 x 8 -videomatriisin sekä 384 x 384 -äänimatriisin sisään- ja ulostuloja. [2.] V_pro8-järjestelmän yksi tärkeimmistä toiminnollisuuksista on purkaa sisääntulevan videosignaalin ääniraidat ja uudelleen summata ne halutuista ääniraidoista ulosmenevään videosignaaliin.

Tarkastellaan NEP:n V_pro8-järjestelmän konfiguraatiota. V_pro8-järjestelmää ohjataan verkkosovelluksen kautta (kuva 2), josta voidaan valita haluttu videokanava, johon halutaan reitittää äänikanavia. Jokainen videokanava tukee 16 yhtäaikaista ääniraitaa.

Kuva 2. V_pro8-videoprosessorin käyttöliittymä.

Videokanavalle voidaan valita ääniraitoja muiden videomatriisissa olevien videokanavien ääniraitojen sisääntulevia signaaleja tai valita MADI-in ääniraitojen sisääntuloja. MADI-In-ääniradat ovat äänimatriisissa olevia ääniraitoja, joihin voidaan esimerkiksi miksata yhteen Lawo-yrityksen crystal-miksauspöydällä selostajien sekä videomatriisiin sisääntulevan TV-lähetyksen kansainväliset ääniraidat. [3.]

Edellä esitellyssä kuvassa 2 V_pro8-videoprosessorin uloslähtevän VPRO_A1-linjan ääniraidan pareihin 1&2 sekä 3&4 on matrisoitu MADI-In-ääniraitoja. VPRO_A1-linjan uloslähteviin ääniraitoihin 5&6 on matrisoitu VPRO_A1-linjan sisääntulevat ääniraidat 1&2. VPRO_A1-linjan uloslähteviin ääniraitoihin 7&8 on matrisoitu VPRO_A1-linjan sisääntulevat ääniraidat 3&4.

V_pro8 tarjoaa kaiken tarpeellisen haastavimpiinkin sekä laajempiinkin playout-järjestelmiin. Sen kattavuudessa on myös heikot puolensa, sillä kaikki monipuoliset mahdollisuudet, mitä tuote tarjoaa, luovat myös mahdollisuuden virheellisiin kytkentöihin. Jouhevan verkkosovelluksen ansiosta käyttäjälle ei välttämättä synny samankaltaista tuntumaa ja varmuutta kytkennöistä kuin fyysisten kaapelien kytkennässä.

3.3 EVS XS -palvelin

EVS-palvelimet ovat DOS-pohjaisia videopalvelimia, joiden konfigurointiin ja kontrolloimiseen on myös tarjolla etäkäyttöyhteydellä toimiva graafinen käyttöliittymä IPDirector. EVS-palvelimet olivat alun perin TV-lähetyksen toimittamiseen suunniteltuja järjestelmiä. Nykyään niiden rooli modernissa TV-tuotannossa on enemmänkin live-lähetysten hidastuskuvien toistaminen. Tästä huolimatta EVS-palvelimista löytyy useita hyödyllisiä toimintoja live-lähetyksiin, videomateriaalin käsittelyyn ja arkistointiin.

Tarkastellaan NEP:n EVS XS2 -palvelimen konfiguraatiota. Yksi palvelin tukee kahdeksaa kanavaa, joista voidaan valita kuinka monta nauhoittavaa Rec-kanavaa käytetään suhteessa PGM-kanaviin eli ohjelmakanaviin. Ohjelmakanavista voidaan kontrolloida haluttuja Rec-kanavia. Rec-kanavat nauhoittavat jatkuvasti niihin matrisoitua lähdettä kirjoittaen aina vanhimman osion yli. Kun palvelimella on käytössä kuusi tallentavaa kanavaa, on tilaa keskimäärin kahdeksan tuntia per nauhoittava kanava. Palvelin allokoi tilaa tasaisesti kanavien välillä välttääkseen tilannetta, jossa yhden kanavan täytyttyä muut kanavat olisivat täysin vapaita. Videomateriaalin tallentaminen vie palvelimelta tilaa näin lyhentäen nauhoitettavaa aikaa. Useita EVS-palvelimia saadaan keskustelemaan keskenään ja ne voidaan konfiguroida käyttämään toistensa kanavia.

EVS XS2 -palvelimen Rec-kanaviin on reititetty V_pro8-järjestelmän uloslähtevät videokanavat. Videomateriaali voidaan leikata käyttäen EVS XS2 -palvelimen graafista käyttöliittymää IPDirectoria (kuva 3).

Kuva 3. EVS-palvelimen graafinen käyttöliittymä, IPDirector. [4.]

EVS XT2 -palvelimen IPDirector-käyttöliittymällä voidaan hallita sekä leikata palvelimella olevaa videomateriaalia. Ohjelmakanavilla voidaan toistaa soittolistaa leikatusta tai palvelimelle siirretystä materiaalista.

Leikkaamisessa voidaan käyttää erilaisia ohjaimia kuten BEPlay (kuva 4) tai Multicam. Ohjaimiin voidaan ohjelmoida halutut toiminnot. Yksinkertaisimmillaan videomateriaalin leikkaus vaatii IN ja OUT -pisteiden määrittelemisen Rec-kanavalla nauhoitetun alueen sisällä sekä OUT-komennon videoklipin tallentamiseksi.

Kuva 4. EVS-palvelimen BEPlay-apuohjain. [5.]

BEPlay-apuohjainta voidaan käyttää apuna videomateriaalin leikkaamisessa. Ohjaimeen voidaan ohjelmoida haluttuja leikkaamista helpottavia toimintoja.

Videoklippi nimetään järjestelmällisesti ja lähetetään FTP:tä (engl. File Transfer Protocol) hyödyntäen XSquare-liitännäisellä NAS:ille, josta se on valmiina ajettavaksi lähetysvirtaan. [6.] Videomateriaalia voidaan nauhoittaa myös Ingest Scheduler -ohjelmistolla. Tällöin EVS XS2 -palvelin tallentaa matrisoitua lähdettä halutusta kellonajasta joko asetettuun kellonaikaan, tai kunnes nauhoitus katkaistaan, ilman että palvelin kirjoittaa videomateriaalin ylitse. Kun materiaali on siirtynyt NAS-palvelimelle, voidaan videomateriaali poistaa EVS XS2 -palvelimelta tilan vapauttamiseksi.

EVS XS2 -palvelimen ohjelmakanavat ovat myös matrisoitu OASYS-järjestelmään, josta videomateriaali voidaan tarvittaessa ajaa lähetysvirtaan. Ohjelmakanavat on myös matrisoitu YLE:n kytkentäkeskukseen meneville kuitupäätteille, joita voidaan hyödyntää videomateriaalin toimituksessa asiakkaille. Kytkentäkeskus vastaanottaa NEP:n videosignaalia ja lähettää sitä edelleen.

3.4 NAS-tallentaminen

EVS XS -palvelimella leikattu videomateriaali siirretään lähiverkon ylitse NAS-palvelimelle, joka on toteutettu Windows 2012 R2 -palvelimena. NAS-palvelin on yhteydessä sekä tarjoaa videomateriaalin OASYS-järjestelmään, joka huolehtii ohjelmalistan järjestyksessä ajamisesta lähetysvirtaan. Videomateriaali, joka on ollut palvelimella määritellyn ajan, siirretään automaatiolla DAC ALTO -palvelimelle käyttäen insinöörityössä tehtyä sovellusta.

3.5 DAC ALTO -arkistointi

ALTO on DAC:in (Disk Archive Corporation) videotuotantoon suunnittelema kustannustehokas arkistointiin suunniteltu palvelin. Se koostuu kehikosta, johon voidaan liittää 48 ALTO-palvelimen tukemaa kovalevyä SATA-III -kaapeleilla, kontrollereista sekä tarvittavista ohjelmistoista. Saatavilla on myös kaksi 60-paikkaista laajennuskehikkoa Mini-SAS -kaapeleilla.

RAID-tekniikan (engl. redundant array of independent disks) sijaan ALTO-palvelin hyödyntää MAID-periaatetta (engl. massive array of idle disks). MAID-periaate pohjautuu suurempaan arsenaaliin kovalevyjä, joista vain tarvittavat levyt ovat aktiivisina. Tuloksena säästetään energiaa ja rasitetaan levyjä vähemmän pidentäen niiden elinikää. [7.]

ALTO-palvelimen ja pääteohjelmien välinen vuorovaikutus käsitellään liitteessä DAC ALTO API (liite 1).

4 Arkistoinnin toteutus

Tässä osiossa tarkastellaan insinöörityössä tehdyn C#-kielellä ohjelmoidun sovelluksen toteutusta. Osiossa käydään läpi sovelluksen oleellisimmat komponentit ja metodit sekä niiden vuorovaikutus. Esiteltävää lähdekoodia on muokattu niin, ettei siitä käy ilmi salassapitosopimukseen kuuluvia asioita.

Pääteohjelman ja palvelimen välinen kommunikointi ja sen toteutus sekä lähdekoodi käsitellään toisessa liitteessä Pääteohjelman ja palvelimen välinen kommunikointi (liite

2). Tiedostonsiirto DAC ALTO -palvelimelle käydään liitteessä Tiedostonsiirto DAC ALTO -palvelimelle (liite 3).

4.1 Sovelluksen asetukset

Sovelluksen asetukset on tallennettu .NET-sovelluksien konfiguraatioille tyypilliseen XML-standardin mukaiseen App.config-tiedostoon (lähdekoodi 1).

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
  </startup>
  <appSettings>
    <add key="FolderPath" value="C:\\Path"/>
    <add key="RemotePath" value="ftp://"/>
    <add key="FTPUserName" value="username"/>
    <add key="FTPPassword" value="password"/>
    <add key="MinDate" value="11/11/2011 11:11:11"/>
    <add key="MaxDate" value="11/11/2011 11:11:11"/>
    <add key="LatestUpdate" value="11/11/2011 11:11:11"/>
  </appSettings>
</configuration>
```

Lähdekoodi 1. App.config-tiedosto.

App.config-tiedostosta ilmenee polku kansioon, mistä tiedostoja siirretään, palvelimen osoite, salasana ja käyttäjänimi FTP-yhteyden todentamiseen, raja-arvot päiville, minkä välistä tiedostoja siirretään ja viimeisin aika raja-arvo mitä ennen tiedostoja on siirretty.

Sovelluksen asetuksia hallinnoidaan AppSettings-luokan (lähdekoodi 2) metodeilla. GetSetting()-metodilla haetaan App.config-tiedostosta avainta vastaava alkio sekä UpdateAppSetting()-metodilla päivitetään avainta vastaavan alkion sisältöä.

```
public class AppSettings
{
    public static string GetSetting(String key)
    {
        try
        {
            var appSettings = ConfigurationManager.AppSettings;
            String result = appSettings[key] ?? "Not Found";
            return result;
        }
        catch (ConfigurationErrorsException)
        {
            ErrorHandling.GenericError();
```

```
            return "Not Found";
        }
    }

    public static void UpdateAppSettings(string key, string value)
    {
        try
        {
            var configFile = ConfigurationManag
            er.OpenExeConfiguration(ConfigurationUserLevel.None);
            var settings = configFile.AppSettings.Settings;

            settings[key].Value = value;

            configFile.Save(ConfigurationSaveMode.Modified);
            ConfigurationManag
            er.RefreshSection(configFile.AppSettings.SectionInformation
            .Name);
        }
        catch (ConfigurationErrorsException)
        {
            ErrorHandling.GenericError();
        }
    }
}
```

Lähdekoodi 2. AppSettings-luokka.

GetSettings()-metodi palauttaa sille annettua avainta vastaavan alkion App.config-tiedostosta (ks. lähdekoodi 1). Alkion sisältö vastaanotetaan String-tyyppiseen merkkijonoon, joka palautetaan metodin kutsujalle. UpdateAppSetting()-metodi vastaanottaa parametreina alkion avaimen sekä sen uuden arvon. Metodi korvaa avainta vastaavan alkion vanhan arvon uudella arvolla.

4.2 Siirrettävät tiedostot

FilesToMove-tietorakenteeseen (lähdekoodi 3) tallennetaan siirrettävän tiedoston nimi, polku sekä tiedoston koko.

```
public class FilesToMove
{
    public String fileName;
    public String directory;
    public long fileSize;

    public FilesToMove(string name, string directory, long size)
    {
        this.fileName = name;
        this.directory = directory;
        this.fileSize = size;
```

```
        }

        public static List<FilesToMove> ListFilesBetweenDates(String path,
        DateTime minDate, DateTime maxDate)
        {
            List<FilesToMove> files = new List<FilesToMove>();
            var directoryInfo = new DirectoryInfo(path);
            int afterMin = 0;
            int beforeMax = 0;

            //Get all files in path
            foreach (var fileInfo in directoryInfo.GetFiles())
            {
                afterMin = DateTime.Compare(fileInfo.CreationTime, minDate);
                beforeMax = DateTime.Compare(fileInfo.CreationTime, maxDate);

                //If between dates add to the list
                if (afterMin > 0 & beforeMax < 0)
                {
                    files.Add(new FilesToMove(fileInfo.Name, fileIn
                    fo.DirectoryName, fileInfo.Length));
                }
            }

            return files;
        }
    }
```

Lähdekoodi 3. FilesToMove-luokka.

ListFile()-metodi lisää FilesToMove-tiedostorakenteeseen kaikki tiedostot sille annetusta osoitteesta aikaväliltä, joka määritellään DateTime-tyyppisten muuttujien MinDate sekä MaxDate avulla. Metodi vastaanottaa String-tyyppisen polun kansioon, josta tiedostoja halutaan siirtää, sekä DateTime-tyyppiset minDate ja maxDate.

4.3 MD5-tiivisteiden vertailu

Ennen alkuperäisen tiedoston poistoa tulee olla varma, että alkuperäinen sekä arkistoitu tiedosto vastaavat toisiaan. MD5-tiivisteitä vertaamalla voidaan todeta tiedostojen olevan samat, jos molempien tiedostojen tiivisteet täsmäävät. Jos tiivisteet eroavat toisistaan, voidaan todeta virheellisen tiedostonsiirron tapahtuneen. Metodi (lähdekoodi 4) vastaanottaa polut alkuperäiseen tiedostoon sekä tiedoston kopioon.

```
        public static bool CompareMD5Hashes(String sourceAddress, String ta
        getAddress)
        {
            //Parse source file MD5 hash
            byte[] sourceHash;
            using (FileStream stream = File.OpenRead(sourceAddress))
```

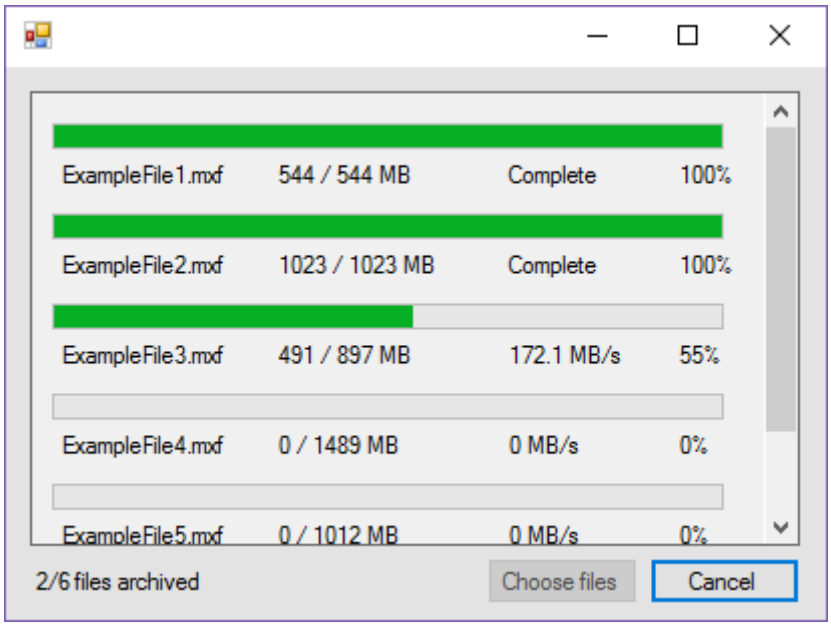
```
        {
            var md5 = MD5.Create();
            sourceHash = md5.ComputeHash(stream);
        }

        //Parse target file MD5 hash
        byte[] targetHash;
        using (FileStream stream2 = File.OpenRead(targetAddress))
        {
            var md5 = MD5.Create();
            targetHash = md5.ComputeHash(stream2);
        }

        //Compare the two hashes
        for (int i = 0; i < sourceHash.Length; i++)
            if (sourceHash[i] != targetHash[i])
                return false;

        return true;
    }
```

Lähdekoodi 4. CompareMD5Comparison()-metodi.

Metodi jäsentelee alkuperäisen tiedoston sekä sen kopion MD5-tiivisteet ja vertaa niitä toisiinsa. Jos yksikin tavu eroaa, palauttaa metodi epätosi-arvon. Jos vertailussa ei havaita tiivisteiden välisiä eroavaisuuksia, palauttaa metodi lopuksi tosi-arvon.

4.4 Käyttöliittymä

Käyttöliittymästä (kuva 5) ilmenee siirrettävät tiedostot, parhaillaan siirtymässä olevan tiedoston arvioitu tiedonsiirtonopeus ja progressio megatavuissa sekä prosentuaalisesti.



Kuva 5. Sovelluksen käyttöliittymä.

Havainnollistetaan käyttöliittymän toiminnallisuutta tiedostonsiirrossa FTP-palvelimelle. Form-luokassa esitellään luokkamuuttujia, joita päivitetään luokan metodeissa. Tärkein luokkamuuttuja on int-tyyppinen currentFileIndex, jonka avulla luokan metodit pääsevät kiinni siirrettävän tiedoston käyttöliittymän elementteihin sekä tietoihin.

4.5 Main-metodi

Main-metodi (lähdekoodi 5) on sovelluksen ydin, jota sovellus kutsuu määritetyin aikavälein tai jota voidaan kutsua käyttöliittymästä manuaalisesti. Manuaalisen kutsun yhteydessä käyttöliittymä pyytää käyttäjää antamaan raja-arvot päivistä, joiden väliltä tiedostoja haetaan.

```
public void Main()
{
    DateTime minDate = Con
    vert.ToDateTime(AppSettings.GetSetting("MinDate"));
    DateTime maxDate = Con
    vert.ToDateTime(AppSettings.GetSetting("MaxDate"));
    sourceDirectoryPath = AppSettings.GetSetting("FolderPath");
    remoteDirectoryPath = AppSettings.GetSetting("FTPAddress");
    userName = AppSettings.GetSetting("FTPUserName");
    password = AppSettings.GetSetting("FTPPassword");

    files = FilesToMove.ListFilesBetweenDates(sourceDirectoryPath, mi
    Date, maxDate);
```

```
        if (files.Count == 0)
        {
            DialogResult dialog = MessageBox.Show("No files to archive",
            "NAS2DAC", MessageBoxButtons.OK);
            if (dialog == DialogResult.OK)
                RefreshClient();

            return;
        }

        InitializeComponents();
        UploadFile();
    }
```

Lähdekoodi 5. Main-metodi.

Metodi hakee sovelluksen DateTime-raja-arvot, polun kansioon, josta tiedostoja siirretään, FTP-palvelimen polun ja käyttäjätunnuksen sekä salasanan yhteyden todentamista varten App.config-tiedostosta (ks. lähdekoodi 1). Sovellus kutsuu ListFiles()-metodia (ks. lähdekoodi 3), joka listaa FilesToMove-tietorakenteeseen DateTime -raja-arvojen sisällä olevat tiedostot. Jos siirrettäviä tiedostoja ei löydy määritellyltä aikaväliltä, käyttöliittymä ilmoittaa siitä käyttäjälle. Ilmoituksen hyväksynnän jälkeen sovellus päivittää käyttöliittymän takasin lähtötilanteeseen. Sovellus luo InitializeComponents()-metodissa ProgressBar-elementin sekä kolme Label-elementtiä jokaista siirrettävää tiedostoa kohtaan. Label-elementeistä ilmenee siirrettävän tiedoston arvioitu tiedonsiirtonopeus ja progressio megatavuissa sekä prosentuaalisesti.

Main-metodi kutsuu alla esiteltyä UploadFile()-metodia (lähdekoodi 6).

```
    public void UploadFile()
    {
        sourcePath = sourceDirectoryPath + @"\" +
        files[currentFileIndex].fileName;
        targetPath = remoteDirectoryPath + files[currentFileIndex].fileName;

        stopWatch.Start();

        client = new WebClient();
        client.Credentials = new NetworkCredential(userName, password);
        client.UploadProgressChanged += new UploadProgressChangedE
        entHandler(UploadProgressChanged);
        client.UploadFileCompleted += new UploadFileCompletedE
        entHandler(UploadFileCompleted);
        client.UploadFileAsync(new Uri(targetPath), sourcePath);
    }
```

Lähdekoodi 6. UploadFile()-metodi.

Metodi lisää siirrettävän tiedoston polkuun FilesToMove-tietorakenteen (ks. lähdekoodi 3) currentFileIndex-luokkamuuttujan osoittaman alkion sisällä olevan tiedoston nimen. Metodi vastaavasti päivittää tämän myös FTP-palvelimen polkuun. Metodi käynnistää StopWatch()-olion, jota hyödynnetään tiedonsiirtonopeuden arvioinnissa (ks. lähdekoodi 7). Metodi luo uuden WebClient()-olion, jolle metodi antaa Main-metodissa (ks. lähdekoodi 5) haetun käyttäjätunnuksen sekä salasanan yhteyden todentamista varten.

Metodi luo WebClient-olion, jolle lisätään UploadProgressChanged sekä UploadFileCompletedEventHandler-tapahtumankäsittelijät (ks. lähdekoodi 7 ja lähdekoodi 8), joiden avulla tiedostonsiirron edistymistä voidaan päivittää käyttöliittymään. Lopuksi metodi aloittaa asynkronisen tiedostonsiirron, jonka etenemistä voidaan seurata alla esitellyssä UploadProgressChanged-tapamahtumankäsittelijässä (lähdekoodi 7).

```
private void UploadProgressChanged(object sender, UploadProgres
ChangedEventArgs e)
{
    double transferSpeed = 0;
    double progressPercent = 0;
    long receivedToMBs = e.BytesSent / 1024 / 1024;
    long totalToMBs = e.TotalBytesToSend / 1024 / 1024;

    if (e.BytesSent != 0)
        progressPercent = ((double)receivedToMBs) / (double)totalToMBs *
        100;

    if (stopWatch.Elapsed.TotalSeconds != 0)
        transferSpeed = receivedToMBs / stopWatch.Elapsed.TotalSeconds;

    progressBars[currentFileIndex].Value = (int)progressPercent;
    percentLabels[currentFileIndex].Text = progressPercent.ToString("0")
    + "%";
    progressLabels[currentFileIndex].Text = receivedToMBs.ToString() +
    " / " + totalToMBs.ToString() + " MB";
    speedLabels[currentFileIndex].Text = transferSpeed.ToString("0.0") +
    " MB/s";

    if (cancelled)
    {
        client.CancelAsync();
    }
}
```

Lähdekoodi 7. UploadProgressChanged-tapahtumankäsittelijä.

Tapahtumankäsittelijä päivittää käyttöliittymään vastaanotetut megatavut, siirtymisen prosenteissa sekä ProgressBar-elementin, kun asynkroninen lataus on edistynyt. Jos

käyttäjä on painanut Cancel-nappia, on bool-tyyppisen cancelled-muuttujan arvoksi vaihtunut tosi. CancelAsync()-metodi palauttaa alla esitellylle UploadProgressComplete-tapahtumakäsittelijä (lähdekoodi 8) e.Cancelled-tapahtuman arvoksi tosi.

```
private void UploadFileCompleted(object sender, AsyncCompletedEventArgs
e)
{
    if (e.Cancelled)
    {
        Console.WriteLine("asd {0}", e.Cancelled);
        progressBars[currentFileIndex].Value = progress
        Bars[currentFileIndex].Minimum;
        speedLabels[currentFileIndex].Text = "Cancelled";
        percentLabels[currentFileIndex].Text = "";
        DialogResult dialog = MessageBox.Show("Archiving cancelled.",
        "NAS2DAC", MessageBoxButtons.OK);
        if (dialog == DialogResult.OK)
               RefreshClient();

        return;
    }

    stopWatch.Reset();
    speedLabels[currentFileIndex].Text = "Complete";
    currentFileIndex++;
    transactionsLabel.Text = currentFileIndex.ToString() + "/" +
    files.Count.ToString() + " files archived";

    if (currentFileIndex < files.Count)
    {
        UploadFile();
    }
    else
    {
        DialogResult dialog = MessageBox.Show("All files successfully
        archived.", "NAS2DAC", MessageBoxButtons.OK);
        if (dialog == DialogResult.OK)
            RefreshClient();

        return;
    }
}
```

Lähdekoodi 8. UploadFileCompleted()-tapahtumankäsitteljiä.

Jos e.Cancelled-tapahtuma on tosi, tapahtumankäsittelijä päivittää käyttöliittymään tiedon tapahtuman keskeytymisestä ja ilmoittaa siitä käyttäjälle. Ilmoituksen hyväksymisen jälkeen sovellus päivittää käyttöliittymän takasin lähtötilanteeseen.

Tiedostonsiirron onnistuessa tapahtumankäsittelijä nollaa StopWatch()-olion seuraavan tiedoston tiedonsiirtonopeuden laskentaa varten ja päivittää käyttöliittymään tiedoston siirtyneeksi. Tapahtumankäsittelijä kasvattaa currentFileIndex-luokkamuuttujaa ja kut-

suu UploadFile()-metodia (ks. lähdekoodi 6) uudestaan, jos currentFileIndex-luokkamuuttuja on pienempi kuin FilesToMove-tietorakenteen (ks. lähdekoodi 3) koko. Jos currentFileIndex-luokkamuuttuja ei ole pienempi kuin FilesToMove-tietorakenteen koko, on kaikki tiedostot siirretty. Tapahtumankäsittelijä ilmoittaa siitä käyttäjälle, jonka hyväksymisen jälkeen sovellus päivittää käyttöliittymän takasin lähtötilanteeseen

Sovellus siirtää lähtökohtaisesti yhtä tiedostoa kerrallaan välttääkseen lähiverkon ruuhkautumisen. Prosessi on hidas, mutta se ei ole kiireinen. Samassa lähiverkossa tapahtuu muuta kriittisempää ja kiireisempää liikennettä, kuten tiedoston siirrot EVS XT2 -palvelimelta NAS-palvelimelle.

5 Yhteenveto ja tulokset

Insinöörityössä tarkasteltiin NEP:n playout-järjestelmän arkistoinnin kannalta oleellisten komponenttien vuorovaikutusta, videomateriaalin eri vaiheita playout-järjestelmässä toimituksesta arkistointiin asti sekä kuinka sen eri vaiheet olivat toteutettu. Työssä käytiin läpi työn tavoitteena olleen sovelluksen toimintaa sekä kuinka se oli toteutettu.

Suurin ongelma, mihin ohjelmiston kehityksessä törmättiin, oli Microsoftin .NET Framework -ohjelmistokomponenttikirjaston File.Copy()-metodin rajallisuus. Metodista ei päässyt käsiksi tiedoston siirron etenemiseen, jota tarvittiin käyttöliittymässä tiedostonsiirron etenemisen havainnollistamiseen käyttöliittymässä. Tämä kierrettiin käyttämällä WebClient-luokkaa, jonka avulla voitiin päivittää tiedostonsiirron etenemistä ja siirtonopeuden arviota käyttöliittymään.

Insinöörityön päätteeksi todettiin, että toteutettu sovellus oli toimiva, ja se täytti työn tilaajan asettamat tekniset määritelmät.

Lähteet

1 OASYS Integrated Playout. Verkkodokumentti. <http://broadstream.com/solutions/oasys/> Luettu 17.5.2017.

2 V_pro8. Verkkodokumentti. <https://www.lawo.com/products/video-processing-core-infrastructure/v-pro8.html> Luettu 17.5.2017.

3 crystal. Verkkodokumentti. <https://www.lawo.com/products/radio-consoles/crystal.html> Luettu 17.5.2017.

4 IPDirector. Verkkodokumentti. <https://evs.com/en/product/ipdirector> Luettu 17.5.2017.

5 BEPlay. Verkkodokumentti. <https://evs.com/en/product/beplay> Luettu 17.5.2017.

6 Xsquare. Verkkodokumentti. <https://evs.com/en/product/xsquare> Luettu 17.5.2017.

7 ALTO Highlights. Verkkodokumentti. <https://www.diskarchive.com/alto-g3/highlights> Luettu 17.5.2017.