Masteroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Det humanistiske fakultet
Institutt for musikk

Anders Bru

# Musikkproduksjon Gjennom VXLAN

## Overføring av flerkanals, ukomprimert lyd over Internett

Masteroppgave i Musikkteknologi
Veileder: Trond Engum

Juni 2020


NTNU
Kunnskap for en bedre verden

Anders Bru

# Musikkproduksjon Gjennom VXLAN

## Overføring av flerkanals, ukomprimert lyd over Internett

Masteroppgave i Musikkteknologi
Veileder: Trond Engum
Juni 2020

Norges teknisk-naturvitenskapelige universitet
Det humanistiske fakultet
Institutt for musikk

NTNU
Kunnskap for en bedre verden

# Abstract

This master's thesis explores the possibility of using commercially available off the shelf technology to conduct multichannel music production over the Internet. The task has resulted in a physical device used for musical collaboration by connecting independent locations over the public Internet. The unit provides seamless and flexible transmission of low-latency, multichannel, full-resolution audio. Through this link, users can easily transmit and receive audio between all connected devices, and collaborate and produce music over the Internet. Similar high quality solutions are currently relatively expensive, and thus not readily available to amateurs and the semi-professional segment. As far as I know, there are no established commercial solutions that correspond to this as of today.

# Sammendrag

Denne masteroppgaven utforsker muligheten for å bruke eksisterende hyllevareteknologi til å gjennomføre flerkanals musikkproduksjon over Internett. Oppgaven har resultert i en fysisk enhet som brukes til musikalsk samarbeid ved å sammenkoble uavhengige lokasjoner over offentlig Internett. Enheten sørger for sømløs og fleksibel overføring av lavforsinkelses, flerkanalslyd i full oppløsning. Gjennom denne koblingen kan brukerne enkelt sende og ta i mot lyd mellom alle påkoblede enheter, og gjennomføre samarbeid og musikkproduksjoner over Internett. Liknende høykvalitets løsninger er foreløpig relativt dyre, og dermed ikke lett tilgjengelig for amatører og det semiprofesjonelle segmentet. Det er så vidt jeg vet ingen etablerte kommersielle løsninger som svarer til dette pr i dag.

# Forord

Denne oppgaven er et resultat av fem flotte år på Musikkteknologi. Jeg vil først og fremst takke min veileder Trond Engum, for motivasjon, oppmuntring og gode samtaler. Stor takk til Otto Wittner ved Uninett og NTNU for tålmodig hjelp under hele prosjektet. Jeg ønsker også å takke alle mine tidligere og nåværende klassekamerater, samt familie og venner, for gode vennskap og minner for livet.

# Innhold

# Figurer

# Kapittel 1

# Introduksjon

Digitale verktøy har forandret måten vi kan samarbeide, lage og utøve musikk på, og er mer tilgjengelig enn aldri før. "My main computer is a laptop" forteller produsent Oak Felder i et intervju med The Verge [1], og viser hvordan han kan skape en full produksjon kun med en laptop og et enkelt lydkort. 21 år gamle Steve Lacy har vist hvordan han produserer og spiller inn låter for verdens største artister på sin iPhone, hvor han kobler instrumenter som gitar og bass rett inn og tar opp i appen Garageband [2]. Gjennom de siste tiårene har det skjedd en enorm teknologisk utvikling som har ført til en endring innen innspilling og produksjon, og vi har gått fra de store studioene, som krevde ressurser for å leie, bruke og ivareta, til 2019s mest solgte album som ble spilt inn og produsert på et soverom [3].

Men selv om verktøyene for produksjon er blitt mindre, billigere og lettere tilgjengelig, og Internett har på mange måter bragt verden tettere sammen, er det enda et steg å gå for å bringe geografisk adskilte studioer og rom sammen på en måte som ligner tradisjonell utøvelse som foregår i samme rom.

Innen områder som kringkasting brukes etablerte løsninger for produksjon over nett. Disse baserer seg på å koble sammen flere lokasjoner, og sende bilde og lyd til et hovedkvarter og kontrollrom, hvor sendingen blir kringkastet fra. For eksempel bruker TV2 i Danmark løsninger fra Nimbra, utviklet av svenske Net Insight, til å gjennomføre sportssendinger [4]. Dette krever minimalt med teknikere på fotballarenaen, fordi video og lyd blir direkte overført fra arenaen og til TV2s hovedkvarter i Odense, hvor redigering foregår og den ferdige sendingen blir kringkastet fra.

Er det mulig å anvende den samme teknologien, men i et system for mindre produksjoner, eller er dette fremdeles teknologi som krever de ressursene store medieselskaper stiller med? Kan de samme prinsippene realiseres i et

mindre oppsett, basert på eksisterende hyllevareteknologi, som når opp til de stramme kriteriene for flerkanalsproduksjon over Internett?

## 1.1 Bakgrunn

Min personlige motivasjon for dette ligger i bakgrunnen min som soveromsprodusent og musiker. Med erfaring fra både hjemmestudio og profesjonelle studioer ser jeg et tomrom innen samarbeid over Internett. For produsenter, låtskrivere og musikere er samarbeid en viktig del av arbeidsprosessen, men det betinger at man enten oppholder seg i samme rom eller bruker fildeling. Løsninger som Skype[1] og Zoom[2] brukes ofte for kommunikasjon, men leverer dårlig lydkvalitet og høy forsinkelse, og er derfor dårlig egnet til musikalsk samarbeid. Et hjemmestudio har ofte begrenset med utstyr og plass, og man må derfor bruke større studio for opptak av diverse instrumenter. Vil man for eksempel gjøre opptak av trommer er en løsning å gjennomføre opptaket lokalt i et studio og deretter sende lydfilene med tjenester som Dropbox til hjemmestudio.[3] Om man ved gjennomlytting av opptakene hører noe som kunne forbedres må hele prosessen gjentas, noe som kan være tidkrevende og frustrerende. En løsning for å sende alle kanalene direkte fra studio, hvor trommene står, til et hjemmestudio ville fjernet det frustrerende mellomleddet.

## 1.2 Motivasjon

Med den enorme utviklingen innen teknologi, og spesielt nettverksteknologi, vil det være naturlig å forvente at det som tidligere var begrenset for større selskaper og systemer nå sprer seg nedover og blir mer tilgjengelig for allmennheten. For kringkastere eller mediebedrifter, for eksempel den danske TV2 produksjonen, er teknologien og vertkøyene tilgjengelig, men dette gjelder ikke for mindre profesjonelle amatører og hobbyister. 86% av norske husstander har tilgang på bredbånd med hastighet på minst 100Mbit/s og 61% har tilbud om 1Gbit/s [5], som burde åpne opp for nye muligheter innen produksjoner over nett.

Samarbeid over Internett har blitt tydeligere aktualisert under koronaviruspandemien, hvor store deler av verden har måttet lære seg å kommunisere over nett. Konserter, bandøvinger og eksamen for utøvende musikere er bare

[1] Skype: https://www.skype.com/no/
[2] Zoom: https://zoom.us/
[3] https://www.dropbox.com/

et par av problemstillingene som raskt måtte løses, og for mange har løsningen vært tjenester som Skype og Zoom. Men disse tjenestene er ikke optimale løsninger for musikksamarbeid på grunn av høy forsinkelse og dårlig lydkvalitet.

Som en del av forarbeidet til oppgaven var jeg med NRK[4] på et utvalg opptak og kringkastinger av Trondheim Symfoniorkester (TSO)[5], gjort i Olavshallen, Trondheim.[6] Dette var for å kartlegge eksisterende praksis og studere hvordan et konvensjonelt opptak innen kringkasting fungerer. Dette innebærer reservering og bruk av en "Lydbuss", en buss gjort om til å bli et musikkstudio på hjul, med de fasilitetene som trengs for å gjennomføre kringkasting. Selv om NRK har lokaler på Tyholt, 1.8km unna, som inkluderer flere skreddersydde studioer, må de bruke Lydbussen, som må stå like utenfor konsertlokalet, for å gjennomføre produksjonen. NRK har kun et fåtall av disse bussene, og i løpet av høsten 2019 ble flere avskiltet, som resulterte i færre opptak av TSO i Trondheim. Et system for flerkanals produksjon over Internett ville ha forenklet oppsettet og ressursene som kreves for å gjennomføre en slik produksjon, og kunne hatt potensiale til å resultere i flere opptak og flere kringkastinger av TSO på radio.

Som en forlengelse av dette forarbeidet intervjuet jeg komponist John R. Graham, som hadde gjennomført et opptak av et orkester i Sofia, Bulgaria. [7] John bor selv i USA og brukte derfor en løsning hvor han mottok en komprimert stereo miks av orkesteret gjennom nettleseren sin og kommuniserte gjennom Skype til teknikere og dirigent i Sofia. Han kunne dermed sitte i sitt personlige studio, lytte til en stereo miks av hele orkesteret og gi tilbakemeldinger om eventuelle endringer i sanntid. John var fornøyd med oppsettet, men det krever fremdeles flere steg før de ferdige lydfilene ender opp på hans maskin. Kvaliteten på lydstrømmen han mottok var ikke høy nok til å gjøre opptak direkte inn i Johns DAW, og han måtte derfor vente på at teknikere skulle gjøre seg ferdige i Sofia, eksportere ut filer og sende de over, for deretter å importere de inn i sitt DAW. *DAW*, kort for *Digital Audio Workstation*, er programvare for produksjon, innspilling og editering av musikk og lyd [6]. Utfra disse erfaringene er det naturlig å spørre: Kan disse ekstra stegene unngås, og vil det være mulig med dagens teknologi å lage et system som kobler to studioer sammen direkte over Internett?

[4]NRK: https://www.nrk.no/
[5]TSO: https://www.tso.no/
[6]Olavshallen: https://www.olavshallen.no/
[7]John R. Graham: http://johngrahammusic.com/

## 1.3 Mål med oppgaven

Med dette spørsmålet som bakteppe ønsker jeg å undersøke om det er mulig å sette opp en enhet for sammenkobling av to studioer, for å kunne gjøre flerkanalsproduksjoner over Internett. For å sette den ferdige enheten og det endelige systemet opp mot eksisterende teknologi er det satt en rekke kriterier som må oppnås. Kriteriene er satt på grunnlag av hva som kreves av et slikt system for å kunne gjøre produksjoner, for at systemet skal bli så enkelt som mulig å ta i bruk, og basert på egne erfaringer fra innspillinger og produksjoner. "Brukeren" vil i denne oppgaven si en lydtekniker, musiker, produsent eller lignende.

Kriteriene er:

- **Multikanal**: Systemet må kunne overføre flere kanaler samtidig og ikke være begrenset av en stereomiks. Basert på erfaringer fra studiopraksis må systemet håndtere minimum 8 kanaler.
- **Ukomprimert data**: For å kunne gjennomføre opptak må lydkvaliteten være så god som mulig, det vil si at dataen ikke skal komprimeres under overføring mellom studioer.
- **Lav forsinkelse**: Oppgaven fokuserer på samarbeid, musikkproduksjoner og opptak. Disse krever lav forsinkelse for å få til god kommunikasjon, men er ikke like tidskritisk som musikalsk samspill. IP-telefoni sikter mot forsinkelse under 150ms for å ivareta samtalekvaliteten, og dette er derfor en veiledende maksimum forsinkelse for dette systemet, men det er naturlig nok ønskelig med lavere.
- **Enkel konfigurering**: Systemet må ikke oppleves som vanskelig å ta i bruk eller vanskelig å integrere i en eksisterende arbeidsflyt.
- **Mobilt**: Enheten skal kunne fraktes uten problemer og få plass i en sekk, slik at man enkelt kan flytte på seg og sette opp en kobling fra et nytt sted.
- **Multi-DAW kompatibelt**: Systemet skal ikke være låst til noen programvare, og brukerne skal kunne anvende hvilket DAW de ønsker i hvert sitt studio uten at det går utover kvaliteten på systemet.

For å måle kvaliteten på systemet vil det være naturlig å sende lyd. I tillegg brukes det to verktøy for å måle data; *Wireshark*[8] og *tcpdump*.[9]. Disse brukes til feilsøking og pakkeanalysering, for å se om all data kommer frem slik den skal.

[8] https://www.wireshark.org/
[9] tcpdump: https://www.tcpdump.org/

## 1.4 Omfang og Limitasjoner

For å fokusere oppgavens innhold til å homhandle produksjon og samarbeid har jeg valgt å ikke gå detaljert inn på nærliggende problemstillinger utenom der det trengs.

Det dekkes ikke videokommunikasjon og det er ikke implementert noen løsning for dette i systemet. Videooverføring er et stort tema, og bringer med seg problemstillinger som båndbredde og kvalitet, som ikke er tatt med i denne oppgaven.

Audio over IP (AoIP) har mange likheter med Voice over IP (VoIP), men de er ikke nødvendigvis samme tema og må heller ikke forveksles. Selv om begge handler om overføring av lyd over nettverk fokuserer de på ulike områder og oppnår naturlig nok forskjellig resultater. Denne oppgaven fokuserer på AoIP og tar ikke for seg problemstillinger innen VoIP og VoIP-tjenester, utenom observasjoner av kvalitet satt opp mot systemer rettet mot AoIP.

## 1.5 Oppgavestruktur

Oppgaven er delt inn i 5 kapitler:

**Kapittel 2 - Eksisterende Teknologi:** Oversikt over relaterte systemer og programvarer, med likheter og ulikheter fra oppgavens mål.

**Kapittel 3 - Bakgrunnsmateriale:** Gir relevant informasjon om teknologi og prosesser for å forstå arbeidet.

**Kapittel 4 - Prosess og Praksis:** Gjennomgang av prosessen og presentasjon av den endelige enheten.

**Kapittel 5 - Diskusjon og Videre Arbeid:** Evaluering og diskusjon av system og tester, og forslag til videre arbeid.

# Kapittel 2

# Eksisterende Teknologi

Dette kapittelet ser på eksisterende løsninger som har likheter med oppgavens mål. Her legges det frem etablerte løsninger for musikkproduksjoner over Internett, og disse blir satt opp mot kriteriene for oppgavens mål. Deretter presenteres AoIP-løsningen jeg har tatt utgangspunkt fra i min implementering; Dante. I tillegg til løsninger for AoIP-produksjoner er det et bredt utvalg kommunikasjonsløsninger over nett, som Skype og Zoom, samt en rekke dedikerte telekonferansesystemer, men disse er utviklet for tale-og videokommunikasjon, og ikke musikk og lydopptak. Dette innebærer få kanaler, lite fleksibel ruting, komprimering av lydsignal, høy forsinkelse og låst programvare, og vil derfor ikke fokuseres på i denne oppgaven.

Audio-over-IP, også kjent som networked audio, hadde på mange måter sin kommersielle start i 1997, med lanseringen av CobraNet [7]. Dette systemet ble utviklet av amerikanske Peak Audio, som var den første kommersielle leverandøren av Audio-Over-IP, og ble brukt i blant annet fornøyelsesparker og konferansesentre som en måte å sende lydsignaler til spesifikke områder. Gjennom årene har det dukket opp flere leverandører, som EtherSound[1], Livewire[2], Ravenna[3] og Dante[4]. Disse produktene leverer høykvalitetslyd på et lokalt nettverk (se kapittel 3.1.3 for utdypning om lokalt nettverk), men er ikke designet for bruk over det offentlige Internett. De senere årene har det derimot kommet en del produkter som er rettet mot produksjoner over Internett. Videre utdyper jeg om tre utbredte løsninger for musikkproduksjoner over nettverk, og sammenligner deres design med kriteriene satt for denne oppgaven; Source-Connect, Nimbra og JackTrip.

[1]Ethersound: https://www.allen-heath.com/ahproducts/ethersound/
[2]Livewire: https://www.telosalliance.com/Axia/Livewire-AoIP-Networking
[3]Ravenna: https://www.ravenna-network.com/
[4]Dante: https://www.audinate.com/meet-dante

## 2.1 Over Internett

### Source-Connect

Source-Connect, utviklet av australske Source Elements, er en kommersiell løsning for opptak over Internett.[5] Med en rekke naboprodukter har de utviklet en serie løsninger for de aller fleste scenarioer, fra lyd-og-video strømming i Source-Live til overføring av filer og filhåndtering med Source-Zip. Deres flaggskip Source-Connect gjør det mulig å gjøre fjernopptak i høy kvalitet og brukes i en rekke studioer, blant annet i Øst-Europa hvor studioer og orkestre spesialiserer seg på nettopp fjernopptak [8]. Under lytting og opptak komprimeres signalet, som minsker datamengden, og man har i etterkant mulighet for å erstatte de komprimerte opptakene med ukomprimerte lydfiler gjennom funksjonen Auto-Replace [9]. Dette betyr at under lytting hører man en lavere kvalitets lydstrøm, men man kan likevel hente høykvalitetsfiler i etterkant. Source-Connect har tre ulike versjoner; *Standard*, *Pro* og *Pro X*, hvor *Pro X* kan gi opp til 6 kanaler, mens *Standard* og *Pro* har maks 2 kanaler [10]. Som nevnt innledningsvis har jeg satt et kriterie på minimum 8 kanaler, og Source Connect vil derfor ikke være velegnet for mitt system.

### Nimbra

Nimbra er en løsning rettet mot høykvalitets mediatransport over IP for kringkasting, og kan sende både lyd og bilde.[6] Det omfatter video og lydproduksjon, og har en rekke funksjoner innen video, lyd-og nettverksteknologi for å gjøre ukomprimerte, flerkilde sendinger. Et av salgspunktene for Nimbra er deres fleksible løsninger med tanke på redundans og pakketap for å kunne gjøre live-sendinger med høy kvalitet (begrepet pakketap forklares nærmere i kapittel 3). Systemet baserer seg på å sende video og lyd til et kontrollrom hvor materiale blir redigert og sendt ut på luften. Dette er et fullskala system for de største kringkastingsproduksjonene, og med en tilhørende prislapp. Fordi det retter seg mot nasjonale og internasjonale kringkastingsselskaper, som har store ressurser og budsjett, brukes det ofte dedikerte fiberkabler for sømløs overføring og kontroll av data [4]. Det betyr at mye av dataoverføringen foregår på et eget nettverk og ikke over det offentlige Internett. Dette vil fjerne en rekke problemområder som denne oppgaven må forholde seg til med tanke på videresending av data over Internett. Det er derfor vanskelig å si hvor mye av Nimbras suksess som kommer av lyd-og-video løsningene og hvor mye som kommer av de dedikerte kablene.

[5] Source-Connect: https://source-elements.com/products/source-connect
[6] Nimbra: https://netinsight.net/products-and-services/nimbra/

### JackTrip

Utviklet av Juan-Pablo Cáceres og Chris Chafe for Stanford University, JackTrip er spesielt utbredt i akademiske kretser, ofte med fokus på samspill. [7] JackTrip er spesifikt utviklet for lydoverføring over Internett, og sender ukomprimerte lydstrømmer direkte mellom påkoblede datamaskiner. Det er ingen begrensning på antall kanaler man kan sende, utenom nettverkets kapasitet og båndbredde, og det er også satt opp for ruting mellom ulike lydkilder, samt inn/ut av DAW [11]. Dette betyr at den treffer på flere av kriteriene for denne oppgavens mål. Der hvor den derimot ikke når opp er på oppsett og konfigurering. JackTrip kjøres fra kommandolinjen i operativsystemet, og har ikke et dedikert grafisk brukergrensesnitt. Dette er et lite intuitivt oppsett, spesielt for lydteknikere som ikke nødvendigvis har mye erfaring med kommandolinjen, og det kan oppleves tungvint å sette opp. Når en kobling er etablert har man ikke mulighet til å endre på instillinger som sample rate, buffersize eller antall kanaler uten å måtte ta ned koblingen og starte på nytt, som kan virke frustrerende for brukeren. Ruting til og fra programmer kan virke komplisert, hvor blant annet oppstarts rekkefølgen av de forskjellige programmene er vesentlig for å få det til å virke. JackTrip er derfor ikke optimalt for dette prosjektet.

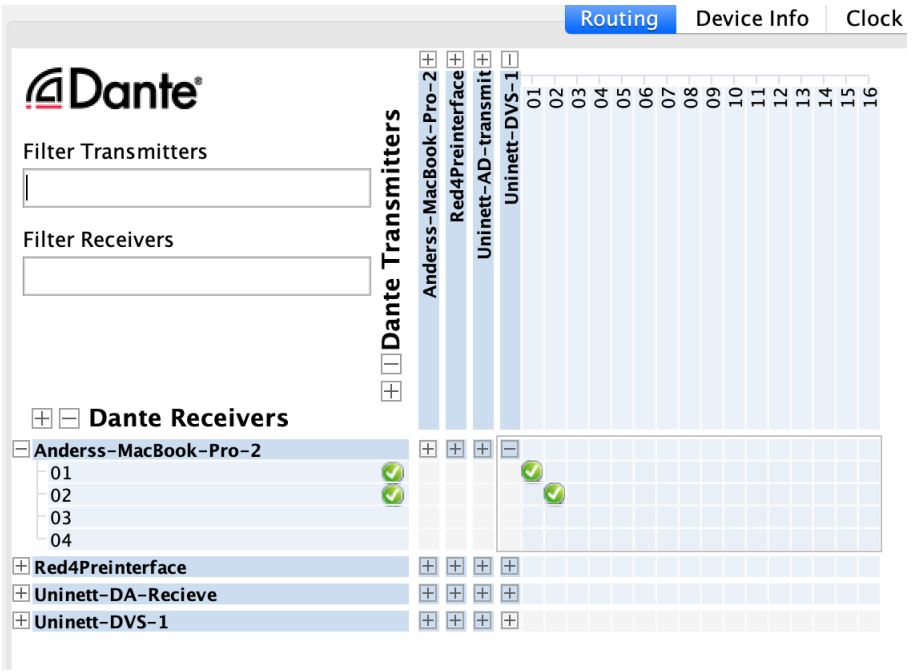
### Andre programvarer

Det er i tillegg en rekke andre løsninger og mindre programmer som omfatter samarbeid og samspill over Internett, men som ikke er velegnet for dette prosjektet. **LOLA** er en lyd-og-video overføringsløsning for samspill over nettverk, men er et komplekst system som krever spesialisert hardware.[8] **UltraGrid**, utviklet av det tsjekkiske forskningsnettverket CESNET, har likheter med JackTrip, men fokuserer på videooverføring, og ikke flerkanalsproduksjoner for lyd.[9] Andre programmer inkluderer **JamKazam**[10], **Reaper Ninjam**[11], **Steinberg VST Connect Pro**[12] og **SoundJack**[13], men disse er rettet mot samspill og er låst til egen programvare. Flere av disse kommer som et resultat av WebRTC, som er en gratis open-source løsning for sanntidskommunikasjon i nettleseren, men som blant annet komprimerer lyden for å spare båndbredde.[14]

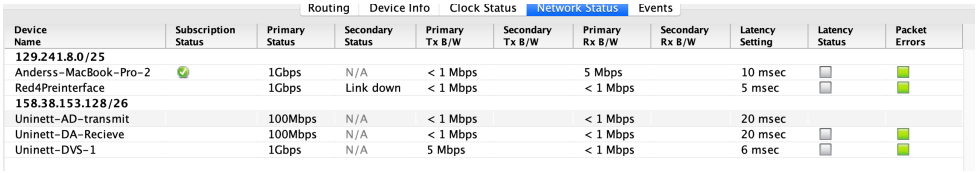
[7] JackTrip: https://ccrma.stanford.edu/software/jacktrip/
[8] LOLA: https://lola.conts.it/
[9] UltraGrid: http://www.ultragrid.cz/
[10] JamKazam: https://www.jamkazam.com/
[11] Reaper Ninjam: https://www.cockos.com/ninjam/
[12] Steinberg VST Connect Pro: https://new.steinberg.net/vst-connect/
[13] SoundJack: https://www.soundjack.eu/index.php
[14] WebRTC: https://webrtc.org/

## 2.2 Dante

Utviklet av australske Audinate, Dante er en løsning for audio over IP på et lokalt nettverk, og inkluderer propriteære protokoller, hardware moduler og software (begrepet protokoll forklares nærmere i kapittel 3). Det er et fleksibelt rammeverk som brukes både i musikkstudio og i større lokaler som konsertarenaer og stadioner for å ha full kontroll over ulike soner og områder [12], og er i dag en av de mest utbredte for AoIP [13]. En stor fordel med å bruke Dante-kompatible produkter er nivået av *plug-and-play*, som vil si at det kreves minimalt med oppsett for å koble sammen enheter. I tillegg til kompatibelt hardware leverer de et virtuelt lydkort, *Dante Virtual Soundcard (DVS)*, som gjør det mulig å koble seg på med en laptop med opp til 64 kanaler inn og ut gjennom datamaskinens interne nettverkskort. Dante kommer med tilleggsprogramvaren *Dante Controller* som viser oversikt og informasjon om tilkoblede enheter, og gjør det enkelt å sette opp ruting mellom enhetene. Denne kombinasjonen av en stor katalog kompatibel hardware, et virtuelt lydkort, en egen kontroller med informasjon om enhetene og høyt nivå av plug-and-play gjør at brukergrensesnittet og fleksibiliteten for oppsett på lokale systemer er ivaretatt. Det har derfor vært naturlig for meg å bruke Dante som mellomledd og signalfordeler inn og ut av forskjellige DAWs i dette prosjektet. Figur 2.1 viser software matrisen til Dante Controller. Figur 2.2 viser informasjonen man får i Dante Controller om nettverket og de påkoblede enhetene, som IP-adresser, båndbredde og latency.




Figur 2.1: Skjermbilde av matrise i Dante Controller.




| Device Name | Subscription Status | Primary Status | Secondary Status | Primary Tx B/W | Secondary Tx B/W | Primary Rx B/W | Secondary Rx B/W | Latency Setting | Latency Status | Packet Errors |
|---|---|---|---|---|---|---|---|---|---|---|
| **129.241.8.0/25** | | | | | | | | | | |
| Anderss-MacBook-Pro-2 | ✓ | 1Gbps | N/A | < 1 Mbps | | 5 Mbps | | 10 msec | | |
| Red4Preinterface | | 1Gbps | Link down | < 1 Mbps | | < 1 Mbps | | 5 msec | | |
| **158.38.153.128/26** | | | | | | | | | | |
| Uninett-AD-transmit | | 100Mbps | N/A | < 1 Mbps | | < 1 Mbps | | 20 msec | | |
| Uninett-DA-Recieve | | 100Mbps | N/A | < 1 Mbps | | < 1 Mbps | | 20 msec | | |
| Uninett-DVS-1 | | 1Gbps | N/A | 5 Mbps | | < 1 Mbps | | 6 msec | | |

Figur 2.2: Skjermbilde av nettverksinformasjon i Dante Controller.

# Kapittel 3

# Bakgrunnsmateriale

Dette kapittelet går videre inn i teknologien og logikken som ligger bak de nevnte systemene - den vil belyse terminologiene og funksjonalitetene som må ligge til grunn for å utviklet et system som innehar kriteriene jeg har satt for enheten jeg har bygd. Kapittelet starter med en oversikt over Internett og nettverk, og en gjennomgang av viktige konsepter. Senere forklares Audio over IP (AoIP) og til slutt tunneleringsteknologien VXLAN, som er brukt for sammenkoblingen over Internett. Dette er et stort fagområde og jeg velger i teksten å kort beskrive de grunnleggende prinsippene før jeg går dypere inn i deler som er mest relevant for oppgaven.

## 3.1 Internett

Internett kalles ofte et nettverk av nettverk. Tre datamaskiner som er koblet sammen vil danne et lite nettverk, og kan sende data seg i mellom. Setter man opp et tilsvarende nettverk i naborommet har man to separate nettverk, og om man kobler det ene nettverket til det andre vil man ha et større, sammenkoblet nettverk, bestående av to mindre. Det er dette som menes med betegnelsen "nettverk av nettverk". Internett er et stort og meget komplekst nettverk, men kan brytes ned på samme måte i mindre nettverk. Det har sitt opphav fra det amerikanske *Advanced Research Projects Agency Network* (*ARPA*, senere kjent som DARPA) på 60-tallet, og har siden det hatt en enorm utvikling.[1] Mange av dagens nettverksprotokoller, som er essensielle for det moderne Internett, var allerede på plass på 70-tallet, men det store gjennombruddet kom ikke før 90-tallet med lanseringen av World Wide Web. I dag er Internett et verdensomspennende nettverk, hvor tjenester som sosiale

[1] DARPA: https://www.darpa.mil/

medier, skylagring og videostrømming står for store deler av datatrafikken [14].

Internett er et *pakkesvitsjet* nettverk (eng: packet-switched). Det betyr at all informasjon som sendes skjer i form av *pakker* [15]. Selve dataen man ønsker å sende puttes inn i en pakke, og disse pakkene sendes fra en avsender til en mottaker, kalt *endesystemer*. Et pakkesvitsjet nettverk er på mange måter likt postvesenet, som også sender pakker fra en avsender til en mottaker, og post kommer derfor til å bli brukt som analogi gjennom dette kapittelet. Dataen (også kalt melding) kan sees på som et brev man ønsker å sende, og pakken fungerer på lik måte som en konvolutt; det er her man skriver informasjon om hvor brevet skal. Postvesenet trenger ikke lese selve brevet for å finne ut av hvor det skal sendes, det står tydelig på utsiden av konvolutten. På samme måte trenger ikke enhetene som sender nettverkspakker (rutere) lese selve dataen (innholdet), men kun hva som står på pakken. Denne prosessen heter *videresending* (eng: forwarding). De to endesystemene i dette prosjektet er to enheter som skal kunne sende pakker mellom hverandre over Internett, og det er på disse enhetene at musikkutstyret (preamper, laptop, mikser, osv.) skal kobles på. Videre forklares hvordan pakkene blir behandlet og sendt over Internett, og hvordan det utnyttes i dette prosjektet for å koble to studioer sammen.

### 3.1.1 Protokoller

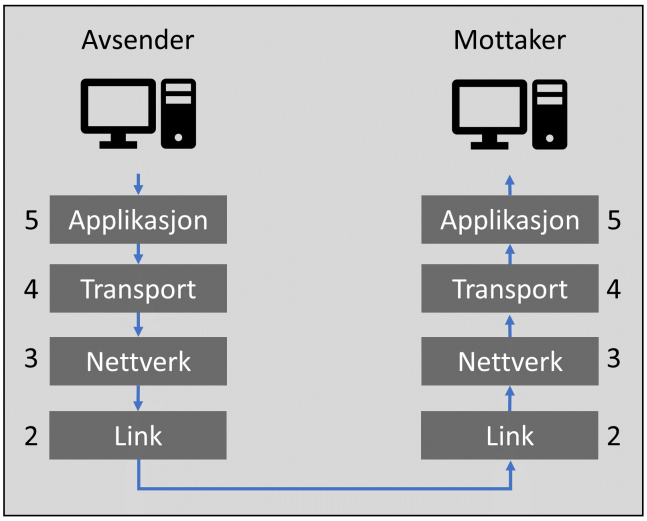
Når man ønsker å sende et brev i posten har Posten satt et par regler for hva som kreves for at de skal sende brevet dit du ønsker. Man kan ikke legge brevet utenfor døren sin og forvente at det dukker opp hos brevvennen din, men brevet må puttes i en konvolutt, det må skrives på navn, adresse og postnummer, samt returadresse. Dette må leveres på postkontoret eller legges i en merket postkasse. Dersom disse reglene følges og informasjonen er skrevet ned slik den skal, kan Posten bruke sitt nettverk av postbud, terminaler og transportmidler for å sende brevet til mottakeren. Men dersom konvolutten ikke har den nødvendige informasjonen vil ikke Posten vite hvor den skal sendes og kan i verste fall ende opp med å måtte kaste hele brevet. På samme måte finnes det regler for hvordan en datapakke skal sendes over et nettverk. Disse reglene kalles for *protokoller* og definerer de syntaktiske og semantiske reglene for kommunikasjon. De beskriver hvordan meldinger skal formateres og hvordan datamaskiner skal håndtere sending og mottak av meldinger. Dersom alle enhetene følger disse protokollene, og endesystemene "skriver på" den riktige informasjonen, vil alle svitsjer og rutere langs veien kunne videresende pakken til riktig mottaker. Dersom protokollene ikke følges vil pakken som regel kastes [16]. Informasjonen som "skrives" på hver pakke legges inn i *headeren*. *Headeren* består av flere felt, og det er her man finner

blant annet avsender-og destinasjonsadresse.

De mest brukte protokollene for vanlig nettbruk er **IP** (Internet Protocol) [17] og **TCP** (Transmission Control Protocol) [18], hvor IP definerer hvordan pakker sendes mellom rutere og TCP definerer hvordan applikasjonene i hvert endesystem håndterer meldinger som blir sendt og mottatt (det finnes to versjoner av IP; IPv4 og IPv6, hvor IPv6 er den nyeste, men IPv4 er fremdeles mest utbredt, og det fokuseres derfor på denne i oppgaven). TCP er en *forbindelsesorientert* protokoll, som vil si at det først etableres en forbindelse mellom endesystemene før pakkene sendes. Dette gir en del tilleggsfunksjonalitet, som gjør at mottaker kan sjekke at alle pakker er kommet frem og at ingen av pakkene er blitt korrupte eller endret under sendingen. Dersom en pakke har falt bort under sendingen, kalt *pakketap*, kan mottaker sende en melding til avsenderen om å resende den tapte pakken. Denne prosessen øker imidlertid den helhetlige forsinkelsen, fordi alle pakker som mottas må "undersøkes", og for overføring av live lyd ønsker man så lav forsinkelse som mulig. For at en pakke skal sendes på nytt må mottakeren oppdage at pakken mangler, sende en melding tilbake til avsenderen om å få tilsendt en ny pakke, og deretter motta den nye pakken. Dette tar lang tid og innen den nye pakken kommer frem er avspillingen forbi punktet hvor pakken skulle vært. Derfor bruker AoIP **UDP**-protokollen [19], som ikke har den samme tilleggsfunksjonaliteten som TCP, og "undersøkelse" av pakkene går dermed raskere. I post-analogien fungerer TCP slik at avsender og mottaker først sender noen "test-pakker" og hører at alt står bra til. Deretter ser mottakeren på hver enkelt pakke som kommer inn, ser at de er uten skade og at de kommer i riktig rekkefølge. Med UDP er det derimot ingen "small talk" på starten, og avsender begynner bare å sende pakker. Mottakeren bryr seg heller ikke om tilstanden til pakkene eller om de kommer i riktig rekkefølge, og dette tar naturligvis kortere tid. Disse protokollene er åpne nettverksstandarder, som betyr at de kan implementeres uavhengig av leverandør.

Protokollene forteller ikke om den helhetlige kommunikasjonen fra avsender til mottaker, men heller om spesifikke områder underveis. Posten har regler for hvordan informasjonen må skrives på konvolutten, hvordan postmannen skal hente brevene, hvordan brevet transporteres til terminalen, hvordan terminalen sender brevet til riktig destinasjon, og så videre. Hvert av disse områdene har sine sett med regler, og det samme gjelder for nettverk. Det baserer seg på *OSI-modellen* (Open Systems Interconnection model), som er en fem-lags modell, hvor hvert lag har sine protokoller (den fullstendige modellen har syv lag, men som regel brukes en forenklet modell med fem lag). Hvert lag fokuserer på én spesifikk *tjeneste*. IP tilhører lag 3, som er ansvarlig for videresending av pakker mellom rutere, mens TCP og UDP tilhører lag 4 og leverer tjenester for applikasjonene hos endesystemene. Når en pakke sendes vil den starte på lag 5 og jobbe seg nedover til lag 1, som er det

fysiske mediumet dataen beveger seg på, som regel kobber-eller fiberkabel. For hvert lag blir det lagt på litt ekstra informasjon, og man sier at dataen blir "pakket inn" i nye lag. Når pakken ankommer mottakeren skjer den samme prosessen i omvendt rekkefølge, og man "pakker ut" dataen. Denne prosessen kalles **inn-og utkapsulering**. Det vil si at en TCP-pakke på lag 4 (kalt *datagram*) blir pakket inn i en IP-pakke på lag 3. Denne blir deretter pakket inn i en lag 2 pakke (kalt *ramme*), hvor Ethernet-protokollen er mest brukt [20]. Figur 3.1 viser hvordan data fra avsenderen blir pakket inn i de ulike lagene, blir sendt over nettverket og deretter pakket ut hos mottakeren.




Figur 3.1: OSI-modellen.

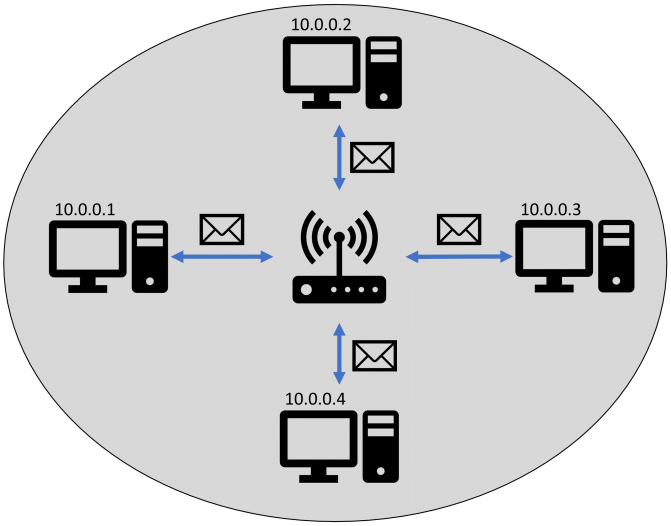
### 3.1.2 Adresser

Om man vil sende en pakke i posten, må man nødvendigvis vite adressen til den man sender til. På samme måte må en avsender av datapakker vite hvor disse pakkene skal sendes. I Norge bruker vi en kombinasjon av postnummer og gateadresse for å definere hvor post skal sendes. Hvert hus har en unik kombinasjon av disse to, og det er derfor enkelt for Posten å videresende pakker til riktig destinasjon. Når en datamaskin kobler seg på et nettverk får den utdelt en slik adresse, kalt **IP-adresse**, og det er denne adressen som brukes til å sende og motta pakker. Avsenderen skriver denne adressen som destinasjonsadresse på pakkene som skal sendes, og rutere på Internett videresender pakkene til mottakeren. Mer konkret så tilhører IP-adressen et spesifikt nettverkskort, slik at en datamaskin med flere nettverkskort koblet på samme nettverk vil ha flere IP-adresser; én til hvert kort. IP-adresser bruker *dotted-decimal notasjon*, som vil si at den bruker fire nummer separert

med et punktum, f.eks. 192.158.0.10. IP-adresser er noe vi ikke behøver å forholde oss til i hverdagen, fordi enhetene vi kobler på Internett får automatisk utdelt en adresse, enten fra en ruter på nettverket eller en dedikert server, kalt *DHCP-server* (Dynamic Host Configuration Protocol) [21]. Fordelen med en automatisk DHCP-tjeneste er at man kan flytte på seg, koble seg på et nytt nettverk og få en ny adresse. De fleste nettverk vi forholder oss til i hverdagen har dette nivået av automatikk, og kalles for *zero-configuration network* fordi det ikke kreves noen manuell konfigurasjon fra brukeren [22]. Den gamle adressen på det forrige nettverket er nå ledig for nye enheter å bruke, og slik kan man spare på adresser. For lydoverføringen kreves det derimot en stabil kobling og applikasjonene må kontinuerlig vite adressene til de påkoblede enhetene. Denne oppgaven skal koble sammen to studioer, og dersom hvert studio bruker hver sin separate DHCP-tjeneste, som deler ut sine adresser, kan det fort bli surr i hvilke enheter som har hvilke adresser. Derfor brukes det *private* adresser som er spesifisert fra brukeren for å påse at ingen enheter får samme adresse og at hver enhet har den samme adressen hver gang den kobles på (som ikke nødvendigvis er tilfelle med en automatisk DHCP-tjeneste).

### 3.1.3 LAN

Dante og liknende tjenester er designet for å fungere på et lokalt nettverk. Det vil si enheter som er koblet sammen, ofte gjennom en svitsj, og som ikke må sende data over Internett for å nå hverandre. Dette kalles et *Local Area Network*, eller **LAN**. I et studio vil ulike enheter være koblet sammen og løsninger som Dante bruker IP-protokollen for å sende pakker (lyd) mellom enhetene. Enhetene har hver sin IP-adresse, men behøver ikke forholde seg til Internett. Prosjektets ferdige system må derfor kunne beholde denne funksjonaliteten og samtidig sende data over Internett. Fordi Dante er designet for LAN, fokuserer systemet på en videreføring og utvidelse av LAN, kalt *VXLAN* (kapittel 3.2 forklarer VXLAN). Figur 3.2 viser et enkelt oppsett med fire datamaskiner koblet på samme svitsj. De har hver sin adresse og kan sende meldinger til hverandre gjennom svitsjen.



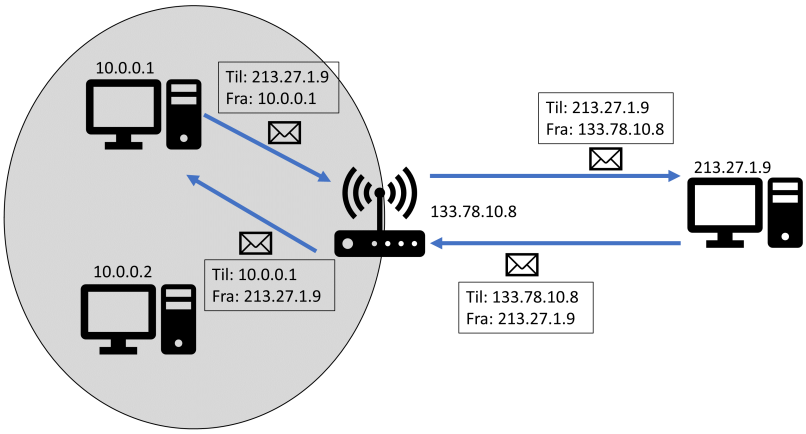

Figur 3.2: LAN med fire datamaskiner koblet på samme svitsj.

### 3.1.4 NAT

For å kunne sende en pakke i posten må man nødvendigvis vite adressen til mottakeren, og for at Posten skal kunne sende pakken må den adressen være unik. Dersom to hus har samme adresse vil ikke Posten vite hvor pakken skal. Som nevnt bruker IP-adresser fire nummer i *dotted-decimal notasjon*. Hvert nummer kan ha en verdi mellom 0-255. Det betyr at det er et begrenset antall unike IP-adresser, og i 2011 ble de siste adressene delt ut [23]. Det vil si at det ikke lenger finnes noen flere unike IP-adresser. Dette problemet ble tidlig identifisert, og det ble innført en løsning for å spare på de unike adressene. Denne løsningen skaper et skille mellom **offentlig** og **privat** IP-adresse. De private adressene er ikke unike, og når man kobler seg på et nettverk får man som regel utdelt en slik privat adresse. Men ruteren man er koblet på har derimot en unik adresse. Det vil si at IP-adressen til ruteren er det ingen andre som har. Dette kan sammenlignes med et leilighetsbygg, hvor alle leilighetene tilhører samme gateadresse, men har unike nummer; Fjellvegen 2B, 2C, 2D osv. Bokstavene "B, C, D" vil være de private adressene, mens "Fjellvegen 2" er den offentlige adressen. Postkassen fungerer som en ruter, hvor all post må først innom den, og blir deretter sortert til riktig leilighet. Det betyr at andre leilighetsbygg kan bruke de "private" adressene "A, B, C" osv, så lenge de har en unik offentlig gateadresse.

Når man kontakter en nettside sender man en forespørsel til en server hvor nettsiden ligger. I forespørselen må man skrive avsenderadresse, slik at serveren vet hvor den skal sende svaret. Men om man er koblet på et lokalt

nettverk, med en privat IP-adresse (som ikke er unik), hvordan vet serveren hvor den skal sende svaret? Dette er som om man skal sende et brev og skrive mottakeradresse "B". Det er jo en rekke hus og leiligheter med adresse "B". Dette løses ved at ruteren på nettverket loggfører din lokale adresse og bytter ut den adressen med *sin* offentlige IP-adresse. Det betyr at når serveren skal svare skriver den inn ruterens offentlige IP-adresse som mottaker. Når svaret ankommer ruteren kikker ruteren i loggen sin, ser at svaret kommer fra en adresse som nylig er blitt kontaktet og sender pakken videre til den lokale datamaskinen. Denne prosessen heter *Network Address Translation*, eller **NAT**. Figur 3.3 viser et eksempel på hvordan avsenderadressen endres når ruteren bruker NAT. En datamaskin på det lokale nettverket sender en pakke til en server, og serveren svarer med ruterens offentlige IP-adresse som destinasjon. Ruteren videresender deretter pakkene til den riktige lokale datamaskinen.




Figur 3.3: Eksempel på NAT.

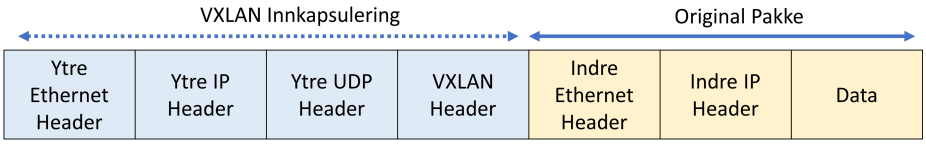
## 3.2 VXLAN

VXLAN er utviklet for bruk innen datasentre, hvor nyere utvikling har bragt med seg større krav om ytelse og fleksibilitet. Mange datasentre bruker *Virtual Machines* (VM), som vil si at en fysisk server (datamaskin) kan ha flere "digitale" servere innebygd. Serverene er påkoblet et stort nettverk og må kunne videresende data raskt og effektivt. I et mindre nettverk kan man bruke *VLAN*, som vil (veldig forenklet) si å lage grupper i nettverket, hvor hver gruppe kun kan sende data til andre enheter i den samme gruppen. Det en måte å forenkle et komplisert nettverk på. Men VLAN er en lag 2 teknologi, og er derfor begrenset til å bruke de protokollene som hører til lag 2. Lag 3 protokoller, spesielt IP, er en mer effektiv måte å videresende

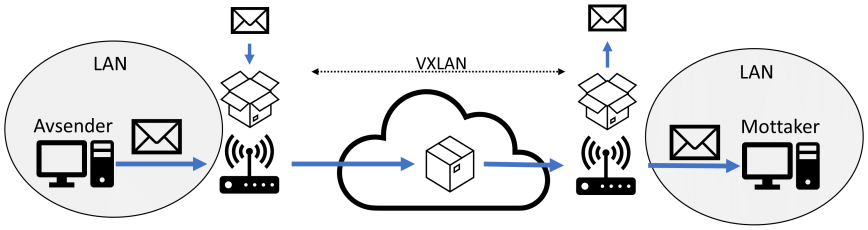
pakker på. Enkelt sagt så trenger man kun fortelle hvor man sender pakker fra og hvor man sender til, og rutere ordner selve transporten på egenhånd (dersom en link går ned eller det er mye trafikk en spesifikk vei, kan IP på egenhånd finne en alternativ rute). Lag 2 protokoller er ikke like selvstendige, og krever mer manuell konfigurering. Og etter som datasentre vokste seg større og implementeringen av VMs ble mer komplisert, krevde det en mer effektiv teknologi enn VLAN.

Cisco, VMware, Arista Networks og flere industriledende utviklere gikk derfor sammen for å utvikle **Virtual Extensible LAN** [24], som åpner opp for å kunne utnytte lag 3. Det vil si å videresende pakker basert på IP-adresser, og ta i bruk rutingprotokoller for å finne raskeste vei mellom servere. Selv om datasentre bruker dedikert hardware for VXLAN, kan det også implementeres i **software**, som er en av hovedgrunnene for valg av VXLAN til dette prosjektet.

VXLAN fungerer som et *overlagsnettverk* (eng: overlay network). Det vil si at man utnytter den fysiske infrastrukturen som allerede eksisterer og bruker VXLAN som et "ekstra" nettverk oppå. På samme måte som at VLAN lagde grupper, lager VXLAN også grupper og man kan kun sende data til de i gruppen. Man sier at VXLAN lager en "tunnel" gjennom nettverket, og hver datamaskin koblet på det VXLANet får et tunnelendepunkt, kalt *Virtual Tunnel Endpoint* (VTEP). I dette tunnelendepunktet legges det på litt ekstra data på alle pakker som blir sendt ut på nettverket, en *innkapsulering*, slik at de blir sendt til mottakere i samme VXLAN gruppe. Denne informasjonen legges på som en ny *header*. Som tidligere forklart innkapsuleres pakker fra høyere lag inn i lavere lag. Med VXLAN legges den ferdig innkapsulerte pakken (som vanligvis ville blitt sendt ut på nettverket) inn i en ny pakke, den blir altså innkapsulert på nytt. Den "ferdige" lag 2 rammen blir innkapsulert i et lag 4 UDP-datagram, og deretter skjer en ny runde med innkapsulering ned til lag 2 igjen. Vi snakker derfor om **indre** og **ytre** pakker; de indre pakkene er den originale lag 2 rammen (som også inneholder de øvre lagene, som er innkapsulert), mens den ytre pakken er den nye informasjonen som blir lagt på i tunnelendepunktet og er kun brukt for transporten over nettverket. Når pakken når frem til tunnelendepunktet hos mottakeren fjernes denne ekstra informasjonen, og den indre pakken sendes videre. Merk at de indre pakkene ikke blir endret under transport over underlagsnettverket. Figur 3.4 viser de indre pakkene og den ytre VXLAN innkapsuleringen. Figur 3.5 viser en forenklet oversikt over hvordan den originale pakken blir innkapsulert ("pakket inn") i en ytre VXLAN pakke, før den blir sendt over Internett til mottakeren, hvor pakken blir utkapsulert og den indre pakken blir videresendt til endepunktet.




Figur 3.4: VXLAN header og innkapsulering.




Figur 3.5: VXLAN Oversikt.

Som forklart i kapittel 3.1.3 operer Dante på et lokalt nettverk (LAN) og bruker lokale IP-adresser. Det vil si at de samme adressene ikke vil fungere over Internett. **Og dette er grunnen til at prosjektet bruker VXLAN for sammenkobling av studio**. Når en pakke blir innkapsulert på nytt i et VXLAN tunnelendepunkt får den **ytre** pakken en ny IP-adresse, uten å endre på den **indre** IP-adressen. Det vil si at den indre pakken (som inkluderer selve lyden man sender) beholder den lokale adressen, og får i tillegg en ny IP-adresse utenpå. Den nye, ytre IP-adressen kan brukes til å sende pakken over Internett. Her setter man mottakerens ruter (med offentlig IP-adresse) som destinasjonsadresse, og rutere langs Internett har ingen problem med å sende den dit. Når pakken ankommer ruteren må man spesifisere hva ruteren skal gjøre med pakken (den har jo nådd destinasjonen sin og ruteren vet ikke hva den skal gjøre med pakken). Her setter man opp *port-forwarding* [25]. Det betyr at man forteller ruteren at alle pakker som kommer inn med ruterens offentlige IP-adresse som destinasjon og et spesifikt *portnummer* skal videresendes til en maskin som er satt opp som tunnelendepunkt inne på det lokale nettverket. Portnummeret er relativt fritt å velge (noen nummer er reservert), men Internet Assigned Numbers Authority (IANA), organisasjon for tildelding og forvaltning av IP-adresser, har tildelt port 4789 som standard for VXLAN og dette burde derfor følges.[2]

[2]IANA: www.iana.org

| External Port | Internal Port | Protocol | To IP Address | Enabled |
|---|---|---|---|---|
| --- | --- | --- | 192 . 168 . 1. 0 | ☐ |
| --- | --- | --- | 192 . 168 . 1. 0 | ☐ |
| --- | --- | --- | 192 . 168 . 1. 0 | ☐ |
| --- | --- | --- | 192 . 168 . 1. 0 | ☐ |
| --- | --- | --- | 192 . 168 . 1. 0 | ☐ |
| 4789 | 4789 | Both | 192 . 168 . 1. 134 | ☑ |
| 0 | 0 | Both | 192 . 168 . 1. 0 | ☐ |
| 0 | 0 | Both | 192 . 168 . 1. 0 | ☐ |

Figur 3.6: Eksempel på port-forwarding konfigurert på en Cisco ruter.

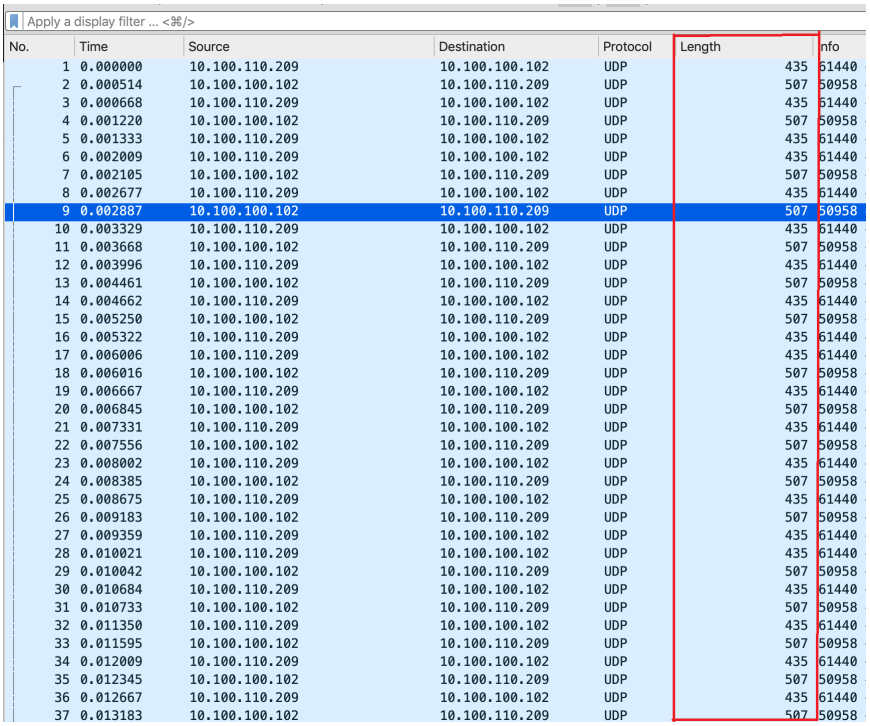
Pakker som går gjennom tunnelendepunktet og blir innkapsulert i en VXLAN pakke blir dermed sendt over Internett til mottakeren, hvor den ytre VXLAN pakken fjernes, og den indre pakken videresendes til riktig enhet. Den indre pakken tror dermed at den fremdeles er på det samme lokale nettverket som den forlot. Som forklart i kapittel 3.1.2 bruker private nettverk de samme adressene, og det er derfor viktig at man har kontroll på IP-adressene til enhetene på begge de lokale nettverkene man kobler sammen, for å unngå at de samme adressene brukes.

### 3.2.1 MTU

For å effektivisere båndbredde er det blitt satt en øvre grense på mengden data, på maksimum 1500 bytes [20]. Denne maksimumstørrelsen kalles for *Maximum Transmission Unit (MTU)*. Dersom en pakke overstiger MTU vil det skje en *fragmentering*. Det betyr at pakken blir delt opp i mindre fragmenter, som blir sendt i hver sin Ethernet-ramme og deretter satt sammen igjen hos mottakeren, kalt *reassembly*. IP tillater alle rutere å fragmentere en pakke dersom det er nødvendig. En funksjon innen fragmentering er at dersom ikke alle de fragmenterte pakkene når frem til mottakeren innenfor et gitt tidsvindu kastes alle pakkene. Dette er selvfølgelig ikke akseptabelt for AoIP hvor hver eneste pakke er viktig, og prosjektets system må derfor kunne holde pakkestørrelsen under 1500 bytes. Under vanlig filoverføring har avsenderen mulighet til å re-sende pakker som ikke når frem, og kan gjør det innen det gitte tidsvinduet. Men under sanntidsoverføring, som AoIP, har man ikke muligheten til å re-sende pakker og disse vil derfor bli kastet.

VXLAN-headeren som blir lagt på pakken bruker 50 bytes med data. Det betyr at dersom den indre pakken bruker full kapasitet (1500 bytes) vil den

totale pakken overstige MTU, og det er fare for fragmentering. Figur 3.7 viser en Wireshark capture av en Dante lydstrøm sendt mellom to enheter. Markert med rødt er pakkestørrelsen til hver enkelt pakke, og den viser tydelig at ingen pakker overstiger 507 bytes. Det betyr at det ikke er fare for fragmentering når VXLAN-headeren blir lagt på.


Apply a display filter ... <⌘/>

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 0.000000 | 10.100.110.209 | 10.100.100.102 | UDP | 435 | 61440 |
| 2 | 0.000514 | 10.100.100.102 | 10.100.110.209 | UDP | 507 | 50958 |
| 3 | 0.000668 | 10.100.110.209 | 10.100.100.102 | UDP | 435 | 61440 |
| 4 | 0.001220 | 10.100.100.102 | 10.100.110.209 | UDP | 507 | 50958 |
| 5 | 0.001333 | 10.100.110.209 | 10.100.100.102 | UDP | 435 | 61440 |
| 6 | 0.002009 | 10.100.110.209 | 10.100.100.102 | UDP | 435 | 61440 |
| 7 | 0.002105 | 10.100.100.102 | 10.100.110.209 | UDP | 507 | 50958 |
| 8 | 0.002677 | 10.100.110.209 | 10.100.100.102 | UDP | 435 | 61440 |
| 9 | 0.002887 | 10.100.100.102 | 10.100.110.209 | UDP | 507 | 50958 |
| 10 | 0.003329 | 10.100.110.209 | 10.100.100.102 | UDP | 435 | 61440 |
| 11 | 0.003668 | 10.100.100.102 | 10.100.110.209 | UDP | 507 | 50958 |
| 12 | 0.003996 | 10.100.110.209 | 10.100.100.102 | UDP | 435 | 61440 |
| 13 | 0.004461 | 10.100.100.102 | 10.100.110.209 | UDP | 507 | 50958 |
| 14 | 0.004662 | 10.100.110.209 | 10.100.100.102 | UDP | 435 | 61440 |
| 15 | 0.005250 | 10.100.100.102 | 10.100.110.209 | UDP | 507 | 50958 |
| 16 | 0.005322 | 10.100.110.209 | 10.100.100.102 | UDP | 435 | 61440 |
| 17 | 0.006006 | 10.100.110.209 | 10.100.100.102 | UDP | 435 | 61440 |
| 18 | 0.006016 | 10.100.100.102 | 10.100.110.209 | UDP | 507 | 50958 |
| 19 | 0.006667 | 10.100.110.209 | 10.100.100.102 | UDP | 435 | 61440 |
| 20 | 0.006845 | 10.100.100.102 | 10.100.110.209 | UDP | 507 | 50958 |
| 21 | 0.007331 | 10.100.110.209 | 10.100.100.102 | UDP | 435 | 61440 |
| 22 | 0.007556 | 10.100.100.102 | 10.100.110.209 | UDP | 507 | 50958 |
| 23 | 0.008002 | 10.100.110.209 | 10.100.100.102 | UDP | 435 | 61440 |
| 24 | 0.008385 | 10.100.100.102 | 10.100.110.209 | UDP | 507 | 50958 |
| 25 | 0.008675 | 10.100.110.209 | 10.100.100.102 | UDP | 435 | 61440 |
| 26 | 0.009183 | 10.100.100.102 | 10.100.110.209 | UDP | 507 | 50958 |
| 27 | 0.009359 | 10.100.110.209 | 10.100.100.102 | UDP | 435 | 61440 |
| 28 | 0.010021 | 10.100.110.209 | 10.100.100.102 | UDP | 435 | 61440 |
| 29 | 0.010042 | 10.100.100.102 | 10.100.110.209 | UDP | 507 | 50958 |
| 30 | 0.010684 | 10.100.110.209 | 10.100.100.102 | UDP | 435 | 61440 |
| 31 | 0.010733 | 10.100.100.102 | 10.100.110.209 | UDP | 507 | 50958 |
| 32 | 0.011350 | 10.100.110.209 | 10.100.100.102 | UDP | 435 | 61440 |
| 33 | 0.011595 | 10.100.100.102 | 10.100.110.209 | UDP | 507 | 50958 |
| 34 | 0.012009 | 10.100.110.209 | 10.100.100.102 | UDP | 435 | 61440 |
| 35 | 0.012345 | 10.100.100.102 | 10.100.110.209 | UDP | 507 | 50958 |
| 36 | 0.012667 | 10.100.110.209 | 10.100.100.102 | UDP | 435 | 61440 |
| 37 | 0.013183 | 10.100.100.102 | 10.100.110.209 | UDP | 507 | 50958 |

Figur 3.7: Skjermbilde av Wireshark.

## 3.3 Audio Over IP

Som nevnt i kapittel 2 har AoIP vært i utvikling i flere tiår. Mange studioer bruker i dag AoIP-løsninger for å koble sammen utstyr, som skaper et fleksibelt oppsett og minimerer kabelbruk. Et slikt oppsett baserer seg på å koble sammen enheter, som preamper, miksere og datamaskiner, til en svitsj, og gjennom en software matrise rute lyd mellom enhetene. Dette bringer med seg en rekke fordeler innen fremtidig skalerbarhet, kostbarhet og "future-proofing" [26]. Men et AoIP-basert system bringer også med seg noen utfordringer, spesielt innen forsinkelse. Analoge signaler må konverteres til digitale og dataen må bli innkapsulert i IP-pakker før det sendes over nettverket, bli pakket ut og konvertert tilbake til analogt signal før det spilles av i høyttalerne. Denne *ende-til-ende forsinkelsen* er avgjørende for et

AoIP system. Voice over IP tjenester som Skype og Zoom kan godta opp til 150ms forsinkelse hver vei, høyere forsinkelse vil gå utover samtalekvaliteten [27]. Innen musikalsk samspill er terskelen betydelig lavere, spesielt for rytmisk samspill hvor det kreves under 14ms forsinkelse mellom musikerne [28]. Merk at den lave forsinkelsen er kun kritisk under samspill, når to eller flere personer skal spille sammen. Under enveis-kommunikasjon, som lag-for-lag opptak, er ikke forsinkelsen lenger kritisk, sett bort fra generell kommunikasjon mellom partene. Om en gitarist står i lokale A og sender livelyd over nett til lokale B, hvor lyden tas opp, har det ikke så mye å si om gitarlyden er forsinket så lenge den kommer helt frem, gitt at gitaristen monitorerer lokalt. Men så fort man introduserer en toveis-kommunikasjon stilles det strengere krav til forsinkelsen.

AoIP krever en stabil kobling mellom alle enheter. I et lukket, lokalt nettverk som ikke er koblet til Internett har man full kontroll over hva som er koblet på og man har en stabil båndbredde. 4 kanaler med 48kHz sample rate bruker rundt 6Mbps båndbredde, som tilsvarer rundt 96Mbps for 64 kanaler[29]. For mange applikasjoner er dette tilstrekkelig med kanaler, og det vil være nok båndbredde i nettverket. Den samme stabile båndbredden får man derimot ikke over Internett, hvor man naturligvis må dele kapasiteten med andre brukere. En ruter på Internett må håndtere stor trafikk, og tidvis kan det bli for mye. De fleste rutere har en buffer hvor datapakker blir lagret før de blir videresendt under stor trafikk, som vil si at pakken blir forsinket eller i verste fall kastet. Dette fører til **pakketap** hos mottakeren og vil høres som et klikk når lyden spilles av. IP-protokollen er en *best-effort* tjeneste, som betyr at den ikke garanterer for levering av pakker eller at pakkene kommer frem innen en gitt tidsramme, og dette kan skape problemer for AoIP. Denne ustabile trafikken kan føre til variasjoner i forsinkelsen, og pakkene vil ankomme mottakeren i uregelmessige intervaller, kalt **Packet Delay Variation (PDV)**, også kjent som **jitter**, som kan skape uønskede artefakter under avspilling. På et lokalt nett er ikke jitter noe problem, både fordi båndbredden er stabil og fordi applikasjoner som Dante har metoder for å kalkulere og fjerne jitter. AoIP-tjenester designet for bruk over Internett, som streaming av lyd og video, telekonferansesystemer, Voice over IP og WebRTC, bruker som regel RTP-protokollen [30], som har metoder for fjerning av jitter og pakketap. Dante bruker derimot sin egen proprietære protokoll som er designet kun for bruk på et lokalt nettverk, og ikke over Internett.

### 3.3.1 Klokker og Synkronisering

Konverteringen fra analogt til digitalt skjer i spesifikke intervaller, kalt *sample rate*. Når den digitale lydstrømmen sendes over Internett til en maskin som

spiller det av må avspillingen skje med de samme tidsintervallene, altså de to enhetene må *synkroniseres*. For å få til denne synkroniseringen brukes det en *klokke*, også kalt *word clock*. Enhetene på nettverket bruker denne klokken til å fortelle i hvilket intervall de ulike pakkene skal avspilles og for å synkronisere flerkanals lydstrømmer. Fordi det digitale signalet må konverteres til analogt i en gitt frekvens er AoIP spesielt utsatt for forsinkelse, og trenger derfor en stabil klokke. RTP-protokollen, som brukes for mediestrømmer over Internett, legger ved et tidsstempel i hver pakke som sendes, mens Dante, som brukes på et lokalt nettverk, bruker **Precision Time Protocol (PTP)** [31]. PTP er en kommunikasjonsprotokoll designet for å synkronisere klokker med nøyaktighet ned på sub-microsekunder, og baserer seg på en master-slave arkitektur [32]. Det betyr at en enhet blir konfigurert som *master* og de resterende som *slaver*. *Master* sender ut PTP-pakker som *slavene* bruker til å synkronisere. Klokkepakkene sendes over *multicast*, som vil si at det sendes flere kopier av samme pakke fra én master til flere slaver samtidig. I et Dante nettverk må klokkemaster være en hardware enhet, *Dante Virtual Soundcard* kan ikke settes til master. Dante bruker adresserommet 224.0.1.129 - 224.0.1.132 for å sende PTP som multicast.

# Kapittel 4

# Prosess og Praksis

Dette kapittelet tar for seg det praktiske arbeidet med å bygge og teste den fysiske enheten som ligger til grunn for problemstillingene. Enheten og systemet må kunne håndtere en rekke kriterier for praktisk bruk, som listet opp i introduksjonen:

- **Multikanal**
- **Ukomprimert data**
- **Lav forsinkelse**
- **Enkel konfigurering**
- **Mobilt**
- **Multi-DAW kompatibelt**

I tillegg er det noen tekniske kriterier:

- **VXLAN i software**: Enheten må ha støtte for VXLAN i software.
- **100 Mbps**: Som forklart i kapittel 3.3 bruker 64 kanaler i 48kHz sample rate rundt 96 Mbps, og dette systemet skal kunne håndtere høyere antall kanaler. Svitsjer har som regel 10/100/1000Mbps båndbredde, og derfor er 100Mbps nærmeste akseptable hastighet.
- **Minimum tre nettverksporter**: Dette er for å koble flere enheter sammen på det lokale nettverket i tillegg til andre enheter som kommer inn gjennom VXLAN-tunnelen.
- **Dante kompatibelt**: Sammenkoblingen over Internett må kunne videresende informasjon som Dante bruker på et lokalt nettverk.

## 4.1 Aruba

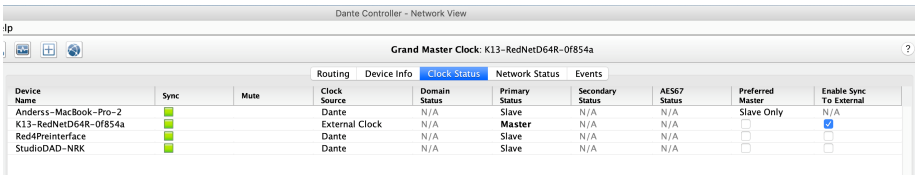
Tidlig i prosjektet ble det brukt en Aruba 2930F JL253A [33], da denne var tilgjengelig fra Institutt for Musikk.[1] Dette er en hardware svitsj, med 24 nettverksporter i tillegg til fiberport, som støtter VXLAN. Aruba svitsjen fungerte bra for formålet og er relativt enkel å sette opp, men det krever en tilkoblet datamaskin. Fordi svitsjen er designet og bygget spesifikt for svitsjing var det ingen problemer med videresending av pakker mellom portene. Den løser derimot ikke alle de satte kriteriene i oppgaven. Med en bredde på 44cm krever den en del plass, i tillegg til at den trenger en ekstern maskin for konfigurering. Prisen i skrivende stund ligger rundt 11000 kroner, som kan oppleves som for dyrt for mange [34]. Med 24 1Gbps nettverksporter har den mer enn nok porter og båndbredde for dette prosjektet, men det går naturligvis utover formfaktoren.


Figur 4.1: Aruba 2930F.

25.april 2019 ble det, som en del av denne oppgaven, gjennomført et opptak av Trondheim Symfoniorkester (TSO) i Olavshallen, i samarbeid med NRK, Uninett og Institutt for Musikk.[2] 24 kanaler ble sendt fra Olavshallen til Tyholt i sanntid over VXLAN. Det ble brukt to Aruba svitsjer som tunnelendepunkt på hver side. I Olavshallen ble mikrofoner koblet til en DAD AX32 preamp, som støtter Dante [35]. Derfra gikk det én enkelt Ethernet-kabel direkte til Aruba svitsjen hvor VXLAN-tunnelen startet. Dette ble sendt over NTNU sitt nettverk opp til NRKs lokaler på Tyholt, hvor det andre tunnelendepunktet var satt opp i et studio. I studio ble det brukt Avid Pro Tools[3] og et Euphonix S5 Fusion miksebord.[4] Det var ønskelig å styre preampene på DAD fra mikseren. Fordi S5-mikseren bruker MADI-protokollen [36] for å

[1] Insititutt for Musikk: https://www.ntnu.no/musikk
[2] Uninett: https://www.uninett.no/
[3] Pro Tools: https://www.avid.com/pro-tools
[4] Euphonix ble kjøpt opp av Avid i 2010: https://www.avid.com/

sende lyd og EUCON-protokollen for kontrollsignal (propritær protokoll fra Avid) ble det brukt en Focusrite RedNet D64R [37] mellom Aruba svitsjen og mikseren, som konverterte fra disse protokollene til Dante. Lydsignalet delte båndbredde med HD-video av lokalet, som ble vist på storskjerm i studio på Tyholt. Det var ingen problemer med pakketap, jitter eller klokkesynkronisering, og latency var på under 1ms. Mikseren kunne også styre input gain på DAD fra Tyholt. Figur 4.2 viser et skjermbilde fra Dante Controller med DAD (*StudioDAD-NRK*), RedNet D64R (*K13-RedNetD64R-0f854a*), samt en personlig MacBook Pro med *Dante Virtual Soundcard* kjørende, hvor alle er synkronisert til RedNet D64R som klokkemaster. Dette viste at man kan sende lyd gjennom VXLAN og at Dante fungerte som det skulle. Men overføringen skjedde over NTNUs lokale nettverk, som fungerer som et stort lokalt nett, og ikke over Internett.




Figur 4.2: Dante Controller

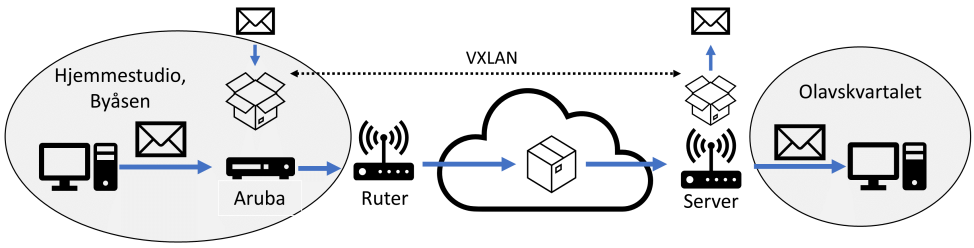
### 4.1.1 Aruba Test

Neste steg var å teste VXLAN over Internett. Det ble gjennomført en test mellom et hjemmestudio på Byåsen og et studio i Olavskvartalet. Aruba svitsjen ble satt som tunnelendepunkt på Byåsen og en server kjørende Ubuntu[5], tilkoblet studio i Olavskvartalet, ble satt som andre tunnelendepunkt.[6] Serveren var tilgjengelig fra instituttet og brukte en statisk, offentlig IP-adresse. Hjemmestudio på Byåsen var bak en hjemmeruter, som betydde at Aruba svitsjen fikk utdelt IP-adresse fra DHCP-tjenesten på ruteren (privat adresse) og fikk ikke offentlig adresse. Serveren i Olavskvartalet kunne dermed ikke kontakte Aruba svitsjen direkte, men måtte komme seg forbi hjemmeruteren først. Grunnet problemer med ruteren var ikke *port-forwarding* vellykket. Det ble derfor brukt en annen løsning, som gikk ut på å sende pakker ut med *Traceroute* fra en laptop innenfra det lokale nettet på Byåsen. *Traceroute* er en innebygd tjeneste i Ubuntu, som lar deg sende pakker til en offentlig adresse, med spesifikk kilde-og destinasjonsport. Dette var viktig fordi gjennom denne tjenesten kan man utnytte NAT-funksjonen til ruteren. Når en pakke går ut fra det lokale nettverket loggfører ruteren hvilken lokale adresse avsenderen har og hvilken port den går ut på. Ruteren venter da på svar fra

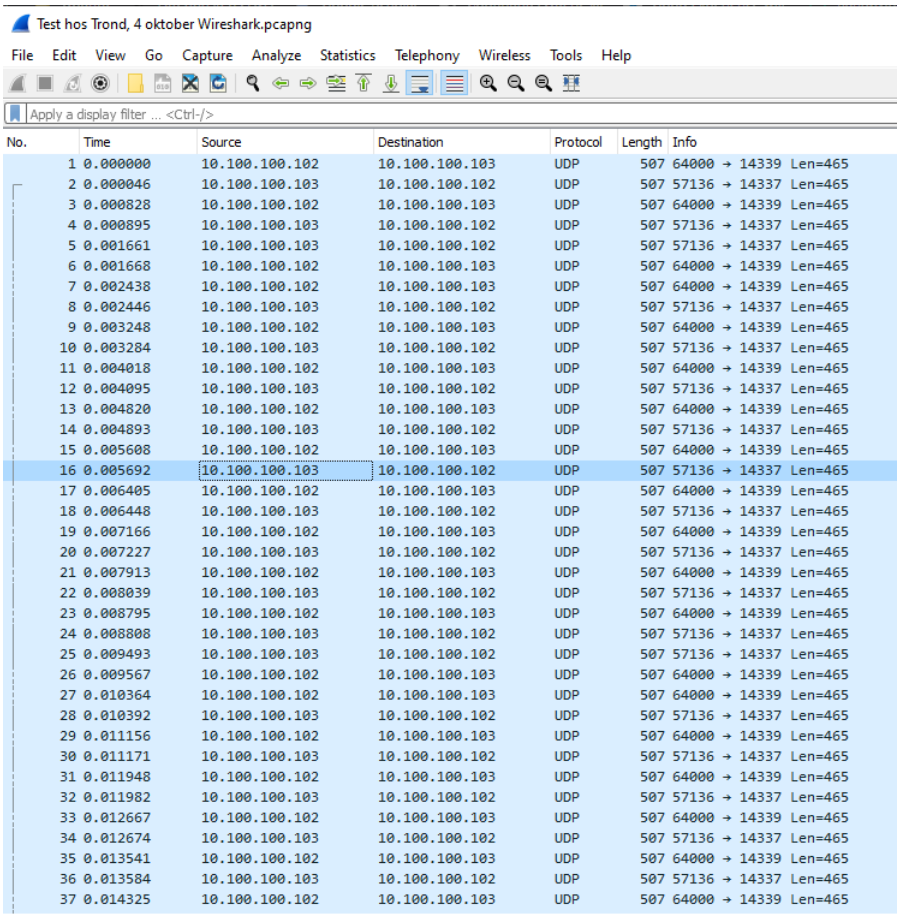
[5]Ubuntu: www.ubuntu.com
[6]Olavskvartalet: https://www.olavskvartalet.com/

IP-adressen pakken var adressert til, med tilsvarende port, og dette kan man bruke til å *åpne* en port på ruteren. Med *Traceroute* ble det sendt pakker til serveren i Olavskvartalet. Når pakker blir sendt andre veien, fra Olavskvartalet til Byåsen, ser ruteren at pakkene kommer fra samme adresse som pakkene tidligere ble sendt ut til, og de har samme destinasjonsport. Den kikker i loggen sin og sender pakkene videre til den lokale adressen som tidligere sendte pakkene ut (med *Traceroute)*. Dersom det ikke hadde blitt sendt *Traceroute*-meldinger fra det lokale nettverket ville pakkene fra Olavskvartalet stoppet hos ruteren på Byåsen, fordi den ikke "venter" på svar fra den adressen og blokkerer pakkene. Figur 4.3 viser en oversikt over signalgangen til testen. Data fra hjemmestudio på Byåsen gikk inn i Aruba svitsjen, hvor den ble innkapsulert og sendt videre til Olavskvartalet.

Det ble koblet på en laptop på hvert nettverk, kjørende *Dante Virtual Soundcard*, og en Focusrite Red 4Pre [38] var påkoblet i Olavskvartalet for å fungerer som klokkemaster (kun hardware enhet som kan være klokkemaster). Testen var delvis vellykket. Figur 4.4 viser skjermbilde av Wireshark, og viser pakker sendt mellom de to laptopene (som bruker statiske IP-adresser). Det vil si at Dante "tror" maskinene er på samme lokale nettverk, selv om de blir inn-og utkapsulert i VXLAN og sendt over Internett. Det ble sendt lyd mellom maskinene, men lyden var ikke helt problemfri. Figur 4.5 viser feilmeldinger i Dante Controller, hvor laptopen på Byåsen med DVS kjørende (*DVSTrond*) hadde problemer med å synkronisere seg til klokkemaster som stod i Olavskvartalet. Dette resulterte i et lydsignal med mye knepping. Vedlegget *"Vedlegg A - Gitar"* er et utdrag fra et opptak gjort i Olavskvartalet, hvor lyden ble avspilt i hjemmestudio på Byåsen. Første halvdel av opptaket har store problemer med knepping, men denne forsvinner halvveis. Gjennom testen kom denne kneppingen inn og ut, og er sannsynligvis påvirket av annen trafikk på Internett. Under perioder med mye trafikk kan det oppstå forsinkelser, noe PTP-pakkene og Dante er veldig sensitiv mot. Testen viste derimot at enhetene så hverandre og fikk sendt data mellom hverandre som om de var på samme lokale nettverk.

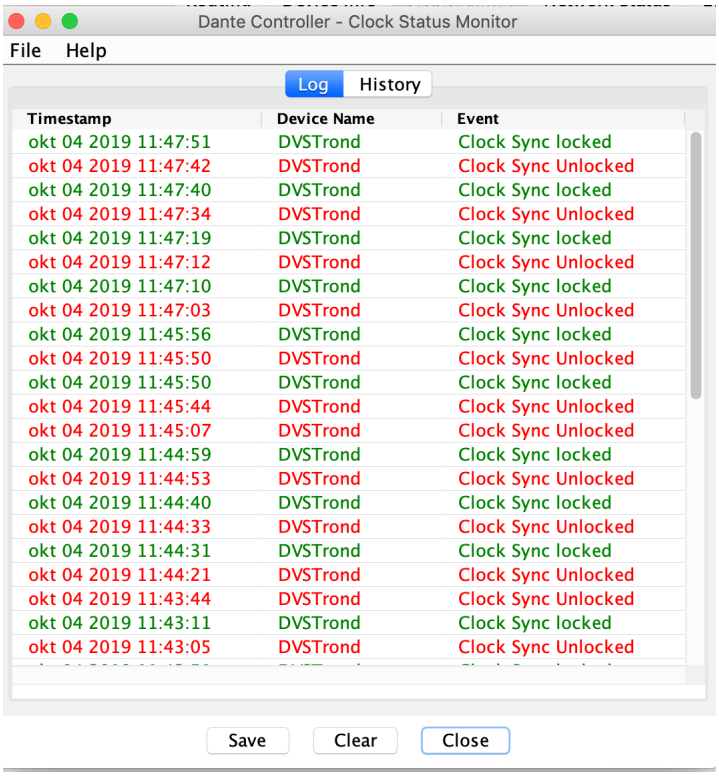



Figur 4.3: Oversikt over Aruba test.




| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 0.000000 | 10.100.100.102 | 10.100.100.103 | UDP | 507 | 64000 → 14339 Len=465 |
| 2 | 0.000046 | 10.100.100.103 | 10.100.100.102 | UDP | 507 | 57136 → 14337 Len=465 |
| 3 | 0.000828 | 10.100.100.102 | 10.100.100.103 | UDP | 507 | 64000 → 14339 Len=465 |
| 4 | 0.000895 | 10.100.100.103 | 10.100.100.102 | UDP | 507 | 57136 → 14337 Len=465 |
| 5 | 0.001661 | 10.100.100.103 | 10.100.100.102 | UDP | 507 | 57136 → 14337 Len=465 |
| 6 | 0.001668 | 10.100.100.102 | 10.100.100.103 | UDP | 507 | 64000 → 14339 Len=465 |
| 7 | 0.002438 | 10.100.100.102 | 10.100.100.103 | UDP | 507 | 64000 → 14339 Len=465 |
| 8 | 0.002446 | 10.100.100.103 | 10.100.100.102 | UDP | 507 | 57136 → 14337 Len=465 |
| 9 | 0.003248 | 10.100.100.102 | 10.100.100.103 | UDP | 507 | 64000 → 14339 Len=465 |
| 10 | 0.003284 | 10.100.100.103 | 10.100.100.102 | UDP | 507 | 57136 → 14337 Len=465 |
| 11 | 0.004018 | 10.100.100.102 | 10.100.100.103 | UDP | 507 | 64000 → 14339 Len=465 |
| 12 | 0.004095 | 10.100.100.103 | 10.100.100.102 | UDP | 507 | 57136 → 14337 Len=465 |
| 13 | 0.004820 | 10.100.100.102 | 10.100.100.103 | UDP | 507 | 64000 → 14339 Len=465 |
| 14 | 0.004893 | 10.100.100.103 | 10.100.100.102 | UDP | 507 | 57136 → 14337 Len=465 |
| 15 | 0.005608 | 10.100.100.102 | 10.100.100.103 | UDP | 507 | 64000 → 14339 Len=465 |
| 16 | 0.005692 | 10.100.100.103 | 10.100.100.102 | UDP | 507 | 57136 → 14337 Len=465 |
| 17 | 0.006405 | 10.100.100.102 | 10.100.100.103 | UDP | 507 | 64000 → 14339 Len=465 |
| 18 | 0.006448 | 10.100.100.103 | 10.100.100.102 | UDP | 507 | 57136 → 14337 Len=465 |
| 19 | 0.007166 | 10.100.100.102 | 10.100.100.103 | UDP | 507 | 64000 → 14339 Len=465 |
| 20 | 0.007227 | 10.100.100.103 | 10.100.100.102 | UDP | 507 | 57136 → 14337 Len=465 |
| 21 | 0.007913 | 10.100.100.102 | 10.100.100.103 | UDP | 507 | 64000 → 14339 Len=465 |
| 22 | 0.008039 | 10.100.100.103 | 10.100.100.102 | UDP | 507 | 57136 → 14337 Len=465 |
| 23 | 0.008795 | 10.100.100.102 | 10.100.100.103 | UDP | 507 | 64000 → 14339 Len=465 |
| 24 | 0.008808 | 10.100.100.103 | 10.100.100.102 | UDP | 507 | 57136 → 14337 Len=465 |
| 25 | 0.009493 | 10.100.100.103 | 10.100.100.102 | UDP | 507 | 57136 → 14337 Len=465 |
| 26 | 0.009567 | 10.100.100.102 | 10.100.100.103 | UDP | 507 | 64000 → 14339 Len=465 |
| 27 | 0.010364 | 10.100.100.102 | 10.100.100.103 | UDP | 507 | 64000 → 14339 Len=465 |
| 28 | 0.010392 | 10.100.100.103 | 10.100.100.102 | UDP | 507 | 57136 → 14337 Len=465 |
| 29 | 0.011156 | 10.100.100.102 | 10.100.100.103 | UDP | 507 | 64000 → 14339 Len=465 |
| 30 | 0.011171 | 10.100.100.103 | 10.100.100.102 | UDP | 507 | 57136 → 14337 Len=465 |
| 31 | 0.011948 | 10.100.100.102 | 10.100.100.103 | UDP | 507 | 64000 → 14339 Len=465 |
| 32 | 0.011982 | 10.100.100.103 | 10.100.100.102 | UDP | 507 | 57136 → 14337 Len=465 |
| 33 | 0.012667 | 10.100.100.102 | 10.100.100.103 | UDP | 507 | 64000 → 14339 Len=465 |
| 34 | 0.012674 | 10.100.100.103 | 10.100.100.102 | UDP | 507 | 57136 → 14337 Len=465 |
| 35 | 0.013541 | 10.100.100.102 | 10.100.100.103 | UDP | 507 | 64000 → 14339 Len=465 |
| 36 | 0.013584 | 10.100.100.103 | 10.100.100.102 | UDP | 507 | 57136 → 14337 Len=465 |
| 37 | 0.014325 | 10.100.100.102 | 10.100.100.103 | UDP | 507 | 64000 → 14339 Len=465 |

Figur 4.4: Wireshark, pakker mellom Byåsen og Olavskvartalet



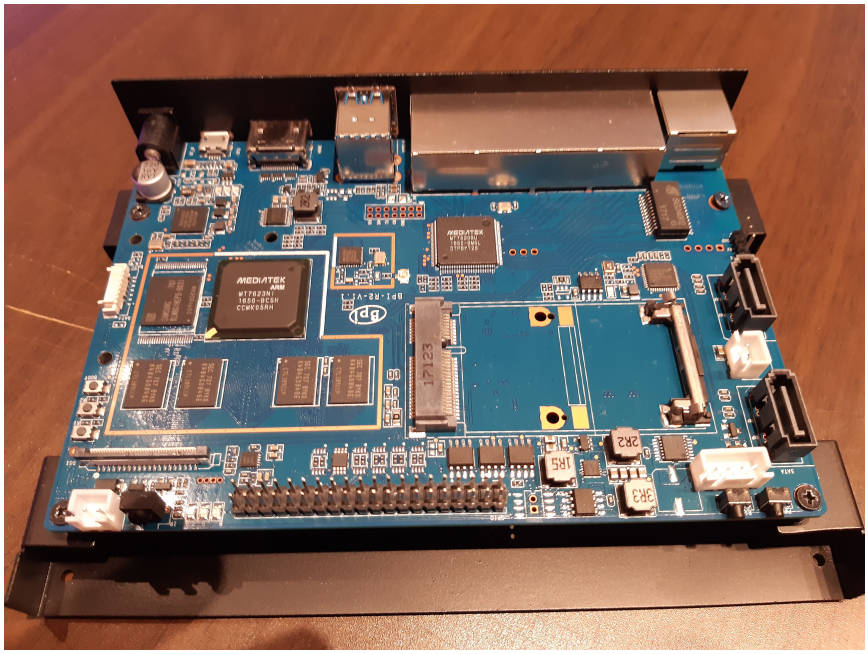

Figur 4.5: Feilmeldinger i Dante Controller

## 4.2 BPI-R2

Neste steg var å se etter en enhet med mindre formfaktor enn Aruba svitsjen. Valg av enhet falt på en *Banana Pi BPI-R2* (BPI).[7] Banana Pi er en serie open source hardware produkter utviklet av kinesiske *GuangDong Bi-Pai Technology*.[8] BPI-R2 er et ruterbasert utviklingskort (eng: development board), som vil si at kortet kun har de nødvendigste komponentene installert og er åpen for videre utvikling og testing. BPI-R2 støtter en rekke operativsystemer, blant annet Ubuntu. Ubuntu er et open source operativsystem som støtter VXLAN og er et naturlig valg for dette prosjektet. Kortet har fem nettverksporter, alle med en overføringshastighet på 1000 Mbps. Portene er fordelt på to nettverkskort, hvor fire porter er på ett kort, markert med "LAN", mens den femte er på et eget og markert med "WAN" (se figur 4.7). *WAN* er kort for *Wide Area Network*, og brukes i denne oppgaven som Internett-tilkobling. Dette betyr at de fire LAN-portene kan fungere

[7]Banana Pi BPI-R2: http://www.banana-pi.org/r2.html
[8]GuangDong: http://wiki.banana-pi.org/Main_Page

som en vanlig svitsj for enheter koblet på det lokale nettverket og at det er en dedikert port for tilkobling til Internett. For implementering i studio er dette viktig fordi BPI-R2 kan settes opp med de lokale enhetene koblet på LAN-portene, og man er alltid koblet på Internett gjennom WAN-porten. BPI-R2 har i tillegg 2GB RAM, som kan være fordelaktig for buffering av data under perioder med stor nettverkstrafikk. Den er 15cm x 10cm stor, som gjør den enkel å frakte med seg (se figur 4.9 for sammenlikning med Aruba), og har et tilhørende deksel som dekker til komponentene. BPI-R2 kan enkelt koble på en skjerm, mus og tastatur gjennom HDMI og USB, og krever ingen proprietære kabler. Den har WiFi-antenne, men dette ble ikke brukt under dette prosjektet fordi AoIP over WiFi ikke anbefales på grunn av høy latency og mye pakketap [39].



Figur 4.6: BPI-R2

Figur 4.7: BPI-R2, bakside med deksel.

Figur 4.8: BPI-R2, framside med deksel.


Figur 4.9: BPI-R2 vs Aruba 2930F

## 4.3 Script

Oppsettet av VXLAN-tunnelen gjøres fra terminalvinduet på BPI-R2. For å forenkle prosessen ble det laget et script som tok imot noen parametere fra brukeren og gjorde resten av oppsettet automatisk (fullstendig script er vedlagt i appendix). Brukeren legger inn den offentlige IP-adressen til mottakeren, portnummer 4789 og deretter hvilket grensesnitt på BPI-R2 som skal brukes. Som tidligere nevnt går WAN-porten på sitt eget nettverkskort og LAN-portene på sitt eget. Grensesnittet man vil bruke som tunnelendepunkt er WAN-porten, som heter *eth1*. Det er her inn-og utkapsuleringen skjer. Deretter lages det en bro mellom WAN-og LAN-portene, slik at alt som kommer gjennom WAN-porten blir videresendt til LAN-portene og omvendt. Det betyr at all data som kommer inn til BPI-R2 gjennom WAN-porten blir utkapsulert og videresendt til LAN-portene, slik at pakkene som ankommer de påkoblede enhetene er de *indre* pakkene, og ikke VXLAN-pakkene. LAN-portene heter *eth0*. Kommandoen for å sette opp tunnelen ser derfor slik ut:

```
1 sudo ./vxlan-setup.sh -r [dstIP] -p 4789 eth1 eth0
```

*sudo ./vxlan-setup.sh* kjører scriptet. *[dstIP]* er den offentlige IP-adressen til mottakeren, og *-p 4789* er portnummeret hos mottakeren. *eth1* er tunnelendepunktet, som vil si den porten som er tilkoblet Internett (WAN-porten). Det siste parameteret, *eth0*, er hvilket grensesnitt broen skal kobles til. I dette tilfellet skal broen kobles på LAN-portene slik at pakker blir videresendt fra tunnelendepunktet til de lokale enhetene.

Ved oppstart av scriptet fjernes eventuelle tunneler som har blitt satt opp tidligere. Dette forsikrer at endepunkter, bro og adresser ikke er i bruk før

man prøver å sette opp en ny tunnel. Delen av scriptet som etablerer tunnelen ser slik ut:

```
1 ip link set $VXLAN down 2> /dev/null
2 ip link del $VXLAN 2> /dev/null
3 ip link add $VXLAN type vxlan id 1 $ADDR dstport $RPORT srcport
      $SRCPORT dev $TDEV
4 ip link set $VXLAN up
```

Linje 1 og 2 tar ned og fjerner tunnelen dersom den eksisterer fra før. Linje 3 setter opp tunnelen med parametere tatt inn fra brukeren. *$ADDR* er IP-adressen til mottakerens tunnelendepunktet, *$RPORT* er portnummeret til mottakerens tunnelendepunkt og *$TDEV* er hvilket grensesnitt tunnelen skal kobles til (*eth1*). Til slutt settes tunnelen opp i linje 4.

## 4.4 Integrasjon med studio

Som nevnt fungerer BPI som en vanlig svitsj for lokale enheter, og den er derfor enkel å integrere i et studio. I kapittel 3.3 er det en gjennomgang av hvordan AoIP brukes i et typisk studio og hvordan enheter som preamper, miksere og PC kan kobles på en svitsj for å rute lyd til hverandre. Når alle enheter er koblet på kan man bruke programvarer som Dante Controller for å rute lyd til og fra enhetene. LAN-portene på BPI skaper ingen hindring for datatrafikk mellom enhetene. Dette gjelder også når BPI er tilkoblet Internett. Det betyr at man ikke trenger å gjøre noen fysiske endringer (omkobling av kabler, oppsett av nytt utstyr o.l.) når man ønsker å sette opp en tunnel, og at alle enhetene som til vanlig er koblet på svitsjen vil også være tilgjengelig for bruk gjennom tunnelen.

## 4.5 BPI-R2 Test

Det siste steget var å se hvordan tunnelen fungerte over Internett og hvor BPI-R2 var bak en hjemmeruter. Da får BPI-R2 en privat IP-adresse som kun fungerer for det lokale nettverket, og ruteren gjør en adresseendring gjennom NAT (se kapittel 3.1.2).

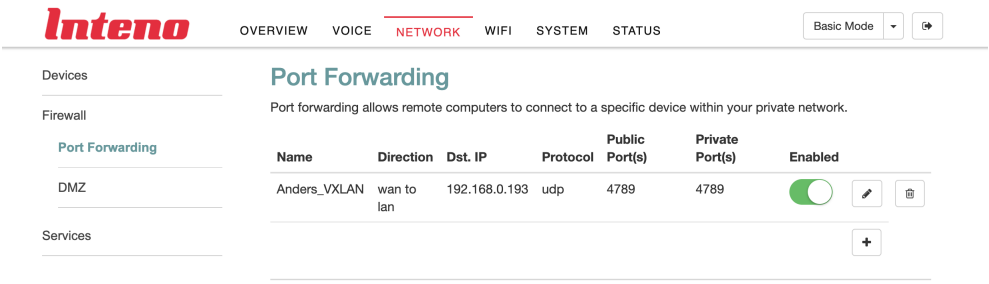
Testen ble satt opp mellom serveren i Olavskvartalet og en stasjonær PC på et hjemmenettverk i Gløshaugveien, 2 km unna. PCen spilte av musikk i Ableton.[9] Til den var det koblet på et *AVIO USB* adapter, som fungerer som et Dante-lydkort.[10] Det betyr at man kan rute lyd til og fra adapteret

[9] Ableton Live: https://www.ableton.com/en/live/
[10] AVIO USB: https://www.audinate.com/products/devices/dante-avio

(to kanaler hver vei), og i Dante Controller vil *AVIO USB* dukke opp. *AVIO USB* var koblet på PC med USB, og gikk direkte til LAN-port på BPI-R2 med en nettverkskabel. Nettverkskabel til hjemmeruteren var koblet til WAN-porten. Vedlegget *"Vedlegg B - Video"* viser både oppsettet i Gløshaugveien og Olavskvartalet. Her ser man den stasjonære PCen i Gløshaugveien, med Ableton spillende (musikken som spilles er selvkomponert), AVIO USB koblet til som lydkort på PC og nettverkskabel koblet til BPI-R2. Man ser også kommandolinjen kjørt fra BPI-R2 for å sette opp VXLAN-tunnelen;

```
1 sudo ./vxlan-setup2.sh -r 129.241.8.10 -p 4789 eth1 eth0
```

IP-adresse *129.241.8.10* er den offentlige IP-adressen til serveren i Olavskvartalet, *4789* er destinasjonsportnummer og *eth1* og *eth0* er grensesnitt for tunnelendepunkt og bro. Fordi PCen var bak en hjemmeruter ble det satt opp *port-forwarding* på ruteren, som vist i figur 4.10. Pakker som kommer inn på ruteren fra Internett, med ruterens offentlige IP-adresse som destinasjonsadresse og destinasjonsportnummer 4789, blir videresendt til BPI-R2 (192.168.0.193, som er den lokale IP-adressen BPI-R2 fikk utdelt fra DHCP).

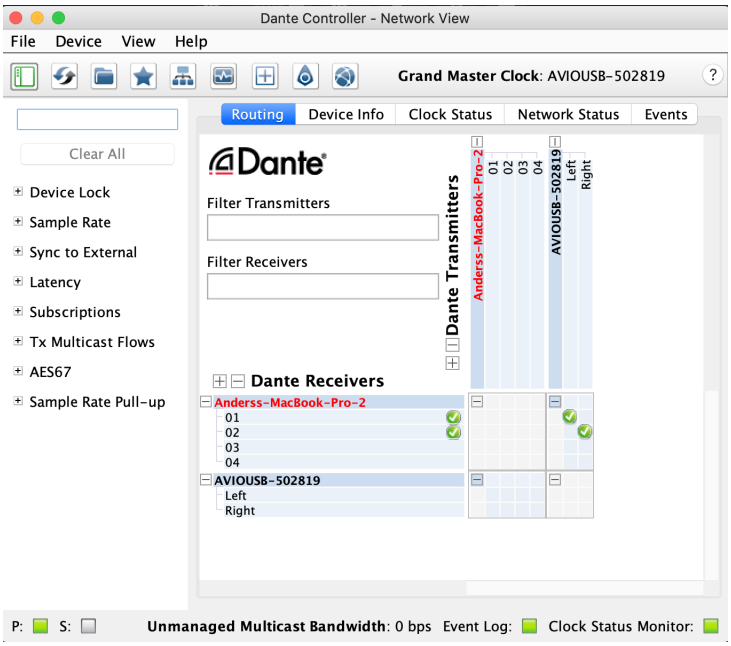



Figur 4.10: Port-forwarding på hjemmenettverk.

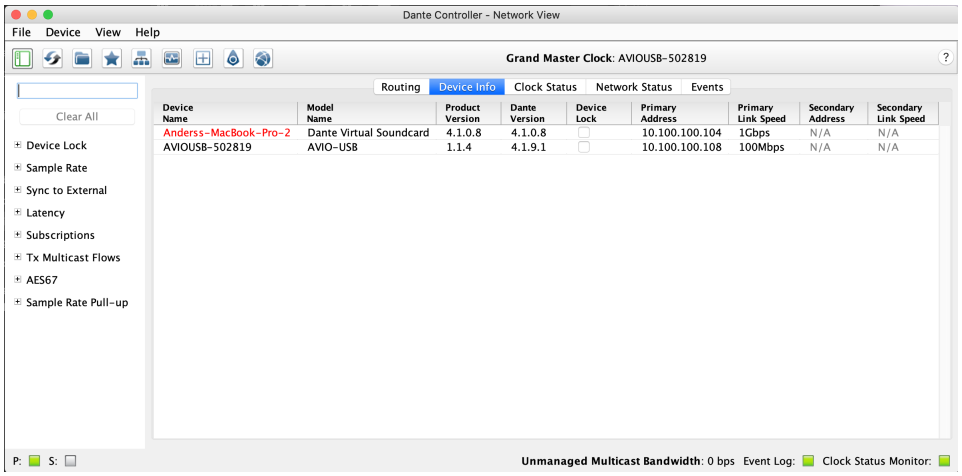
Andre del av den vedlagte videoen *Vedlegg B - Video* viser andre siden av tunnelen, i studio Olavskvartalet. Her ser man serveren som er satt opp som tunnelendepunkt, og som har offentlig IP-adresse, koblet til en svitsj. Svitsjen er for at andre enheter i studio skal kunne koble seg på. For testen ble det koblet på en MacBook Pro, med DVS kjørende. Videoen viser sanntidsoverføring fra Gløshaugveien til Olavskvartalet, monitorert gjennom Ableton og spilt av i høyttalerne i studio. Figur 4.11, 4.12 og 4.13 viser informasjon fra Dante Controller. Her ser man at både MacBook Pro i Olavskvartalet (*Anderss-MacBook-Pro-2*) og AVIO USB i Gløshaugveien (*AVIOUSB-502819*) dukker opp, med vellykket ruting fra AVIO USB til MacBook Pro. IP-adressene i figur 4.12 viser de statiske adressene som er satt manuelt. De viser *ikke* IP-adressen til BPI-R2, hjemmeruteren i Gløshaugveien eller den offentlige IP-adressen til serveren i Olavskvartalet. Altså tror MacBook Pro

og AVIO USB at de er på det samme lokale nettverket.

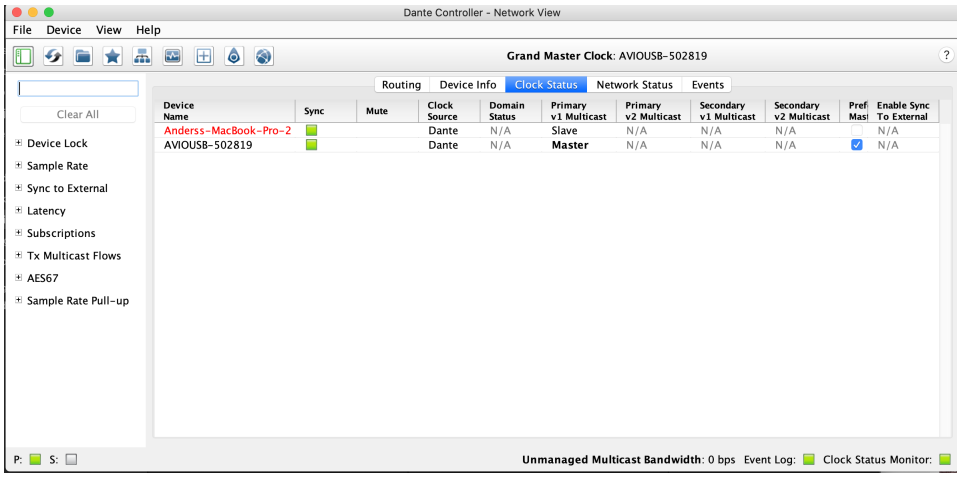
På lik måte som den tidligere testen med Aruba svitsjen var denne testen delvis vellykket. Som vist i videoen greide enhetene å koble seg sammen fra hvert sitt studio og sende lyd over Internett. Figur 4.11 viser matrisen i Dante Controller, hvor både MacBook Pro og AVIO USB dukker opp, og lyd sendes fra AVIO USB til MacBook Pro. *MacBook Pro* er markert med rødt på grunn av en programvarefeil i Dante med *macOS Catalina*. Figur 4.12 viser informasjon om hver enhet, som IP-adressene. Merk at begge enhetene bruker hver sin adresse i 10.100.100.x-serien, da dette er satt manuelt for at enhetene ikke skal få samme adresse utdelt fra DHCP. Figur 4.13 viser klokkesynkroniseringen mellom enhetene. AVIO USB står som *master* og MacBook Pro som *slave*, og bruker PTP-pakker sendt over fra AVIO USB over Internett for synkronisering. Lyden som ble sendt inneholdt tidvis veldig mye knepping. Vedlegget *Vedlegg C - Original* er det originale lydsporet som ble avspilt i Ableton, i Gløshaugveien. Lydklippet *Vedlegg D - Olavskvartalet* er det samme lydklippet sendt over Internett, og spilt inn i Olavskvartalet. Det er tydelige artefakter i første halvdel av klippet, som forsvinner halvveis. Denne kneppingen som oppstår er lik den tidligere testen med hjemmestudio på Byåsen. Den kom med uregelmessige mellomrom, og kunne vare opp til flere minutter før den ga seg. Figur 4.14 viser en *Traceroute* gjort fra Gløshaugveien til Olavskvartalet. Den viser at pakkene videresendes et par hopp før de når Olavskvartalet. Antall hopp skal ikke nødvendigvis være problematisk for tunnelen. Figur 4.15 viser en *Ping*-sesjon mellom Gløshaugveien og Olavskvartalet. *Ping* er et verktøy som brukes for å måle avstand i tid, altså hvor lang tid en pakke bruker på å nå destinasjonen. I de første linjene er det kun et par millisekunder, og dette resulterer i god lyd i Olavskvartalet, uten knepping. Men halvveis øker forsinkelsen veldig, opp til 82 ms. Dette kan komme av stor trafikk langs Internett, som skaper kø hos ruterne. Deretter går forsinkelsen ned igjen til et lavt nivå. Dette "hoppet" i forsinkelse er sannsynligvis årsaken til kneppingen, fordi Dante er avhengig av en stabil kobling.



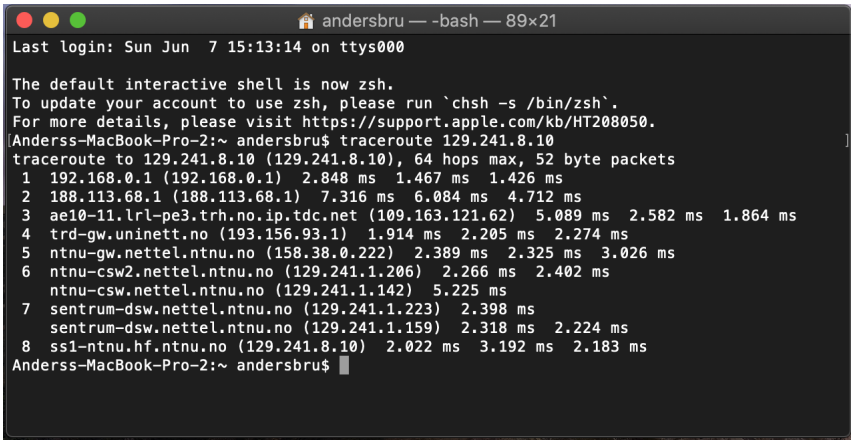

Figur 4.11: Dante Controller, Ruting.



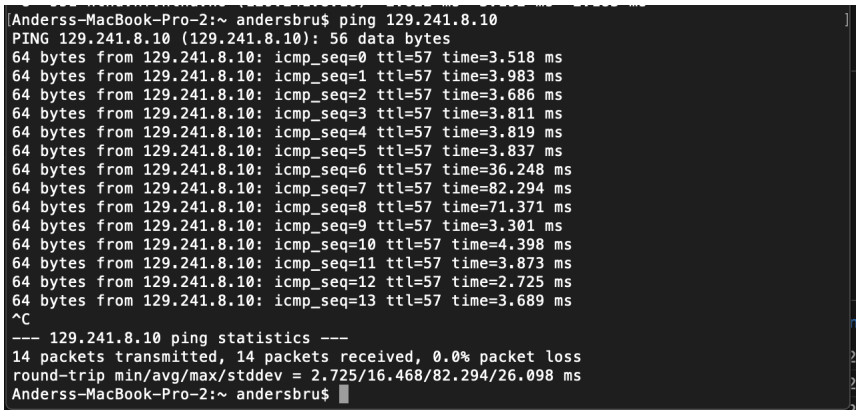

Figur 4.12: Dante Controller, Device Info.




Figur 4.13: Dante Controller, Klokkesynkronisering.




Figur 4.14: Traceroute




Figur 4.15: Ping

# Kapittel 5

# Diskusjon og Videre Arbeid

I introduksjonen til oppgaven ble det lagt frem en rekke kriterier som dette systemet skulle oppnå. Disse ble satt som mål for systemet, både for at systemet skal kunne konkurrere mot eksisterende teknologi, men også for å fylle et tomrom som ikke eksisterer i dag. I dette kapittelet settes den ferdige enheten opp mot disse kriteriene, og det gås dypere inn i hvilke kriterier som er oppnåelige, hvilke som ikke er det, og hvordan systemet fungerer i praksis.

Kriteriene satt for systemet er:

- Multikanal
- Ukomprimert data
- Lav forsinkelse
- Enkel konfigurering
- Mobilt
- Multi-DAW kompatibelt

## Multikanal

Systemet er ikke låst til noen leverandør, men i prosjektets tester ble det brukt Dante. Fordi BPI-R2 har 1Gbps nettverksporter vil man mest sannsynlig ikke bruke opp båndbredden, og restriksjonene på antall kanaler ligger derfor i hardware og hvilken AoIP-løsning man velger. For eksempel kan Dante Virtual Soundcard levere opp til 64 kanaler begge veier, mens Focusrite 4Pre støtter 32 kanaler. Som testene viste er det derimot stor usikkerhet om

hvordan trafikken på Internett er, og flere kanaler kan føre til større sjanse for forsinkelse og pakketap. Sammenliknet med for eksempel *Source Connect* har dette systemet mulighet for flere kanaler (*Source Connect* har begrensning på 6 kanaler), og åpner opp for å koble eksternt hardware sammen.

## Ukomprimert Data

Fordi systemet ikke er låst til noen programvare forholder lyden seg til hvordan den enkelte programvaren velger å løse komprimering. Dante, som ble brukt under testing av denne oppgaven, komprimerer ikke lyden under overføring. Til sammenligning bruker *Source Connect* komprimeringsteknikker under sanntidsoverføring, som vil si lavere lydkvalitet, men med mulighet for å overføre de ukomprimerte lydopptakene som filoverføring i etterkant. Med dette prosjektets system vil man få de ukomprimerte filene rett inn i opptaket hos mottakeren.

## Lav forsinkelse

På et lokalt nettverk kan Dante gi forsinkelse ned på 1ms, gitt at nettverkskomponentene (svitsjene) har høy nok hastighet. Fordi dette systemet ikke fokuserer på samspill er ikke forsinkelsen like kritisk, men det er naturligvis ønskelig med så lav forsinkelse som mulig. Innen IP-telefoni er under 150ms ansett som akseptabelt for å holde en samtale, og dette har blitt brukt som veiledende maksimum forsinkelse for dette prosjektet. Som vist i figur 4.15 var det i hovedsak kun et par millisekunder mellom Gløshaugveien og Kjellerstudio, og selv med et stort hopp (som mest sannsynlig kom fra annen trafikk) holdt forsinkelsen seg under 100ms. Sett bort fra problemer for klokkesynkronisering, var forsinkelsen i seg selv derfor akseptabel for produksjon og samarbeid. Det bør merkes at testene gjort under dette prosjektet har holdt seg i samme by, og det er diskutabelt om man kan beholde den samme forsinkelsen om man øker den geografiske distansen mellom endesystemene.

## Enkel konfigurering

Konfigurasjon og oppsett av tunnelen krevde mye innsats tidlig i prosessen, men har gjennom oppgavens utvikling blitt forenklet. For å kjøre BPI-R2 kreves det svært lite ressurser, kun vanlige datakomponenter som skjerm, mus og tastatur, og oppsett av tunnelen har også blitt forenklet til et script

som brukeren kjører med et par parametre. Sammenliknet med *JackTrip* har dette systemet potensiale for et bedre brukergrensesnitt, dersom man bruker kommersielle løsninger som Dante, Ravenna eller liknende. Ved feilsøking for VXLAN-tunnelen kreves det derimot noe kunnskap om kommandolinjen, på lik linje med JackTrip.

## Mobilt

På grunn av BPI-R2s lille formfaktor er enheten veldig enkel å frakte med seg, og får plass i de fleste bager og ryggsekker. BPI-R2, inkludert ytre casing, er 17cm bred, 4cm høy og 12cm dyp og trenger kun et 12 volts strømadapter for å fungere. Den tar dermed ikke mye plass i et studio, dersom man ønsker å bruke den som en fastmontert enhet for å koble sammen studiokomponentene sine. Figur 5.1 viser BPI-R2 ved siden av en MacBook Pro.

Figur 5.1: BPI-R2 og MacBook Pro.

### Multi-DAW Kompatibelt

Fordi systemet ikke er låst til noen leverandør eller programvare er det også åpent hvilket DAW man bruker. Dante er kompatibelt med alle store DAWs og krever ikke at alle påkoblede enheter bruker samme DAW [40]. Dette gjelder også gjennom VXLAN-tunnelen, som vil si at man på hver side av tunnelen kan bruke forskjellige DAWs.

## 5.1 Konklusjon

Systemet er suksessfullt med å koble sammen to studioer over Internett, og koble eksterne enheter sammen. Hvert studio kan ha eksternt hardware koblet på og enkelt sende lyd mellom enhetene. Systemet er ikke låst til noen programvare, som skaper fleksibilitet hos brukeren. Ved bruk av AoIP-løsninger som Dante kan man sende ukomprimert data mellom studioene, og ivareta den høye kvaliteten man får i et studio. Mange studioer bruker allerede løsninger som Dante for å koble sammen de lokale enhetene, og BPI-R2 vil derfor fungere som en forlengelse av dette. Den lille formfaktoren på BPI-R2 gjør den brukervennlig og enkel å frakte, og det kreves ingen proprietære kabler. Den høye båndbredden på nettverksportene gjør den brukbar for en rekke scenarioer, og med fire LAN-porter kan man koble på fire enheter og samtidig ha en dedikert port til Internett. Fordi oppkoblingen av VXLAN-tunnelen gjøres gjennom et script kreves det kun et par tastetrykk fra brukeren til å sette i gang en kobling, gitt at *port-forwarding* er slått på i den lokale ruteren. Det er derimot noe usikkerhet rundt byggekvaliteten, men den har fungert bra under hele prosjektets periode.

Under prosjektets periode kostet BPI-R2 rundt 1800,- [41]. Sammenliknet med for eksempel Source Connect, hvor billigste løsning er et abonnement på *Source Connect Standard* som koster 324,- per måned og kun gir to kanaler [10], gir dette prosjektets system en mye mer fleksibel løsning. For å få tilgang til seks kanaler med Source Connect må man kjøpe høyeste versjon, *Source Connect Pro X*, som koster 24000,-. BPI-R2 kombinert med Dante gir betydelig mer for pengene med tanke på antall kanaler, lydkvalitet og fleksibilitet.

Men ustabil forsinkelse og pakketap over Internett, samt ingen løsning for re-sending av pakker, gjør at systemet ikke vil fungere godt nok for høykvalitetsopptak. Varierende forsinkelse og perioder med mye trafikk kan problematisere klokkesynkronisering, og føre til knepping i lydsignalet. For at dette systemet skal bli tatt i bruk forventes det ingen problemer med lyden, og selv den minste artefakten kan gjøre et opptak ubrukelig. Under perioder med

lav forsinkelse fungerte det utmerket, men usikkerheten rundt forsinkelsen gjør det vanskelig å faktisk ta i bruk systemet.

## 5.2 Videre Arbeid

Det er mye potensiale for videre arbeid. Én løsning på utfordringene rundt pakketap over Internett har blitt løst av *Source-Connect* og deres *Auto-Replace* funksjon [9]. Under sanntidsoverføring sendes et komprimert signal, som ikke krever like mye båndbredde og dermed minsker sjanse for pakketap. Når mottakeren av signalet starter opptak i sitt DAW startes også et lokalt opptak hos avsenderen. Når opptaket er ferdig kan man enten beholde de komprimerte filene som ble tatt opp, med potensielle klikk etter pakketap, eller velge å bytte ut filene med lydfilene som ble lagret hos avsenderen og naturligvis ikke har noen klikk. Denne prosessen skjer automatisk etter at man velger funksjonen *Auto-Replace* og bytter ut lydfilene direkte i DAW. En liknende funksjon kan implementeres i dette systemet for å motvirke effekten av pakketap.

Fordi Dante er designet for lav forsinkelse bruker den også en lav buffer. Det betyr at den venter en kort tid før den sender de mottatte pakkene til avspilling. Dersom man kan øke denne bufferen vil man kunne ta imot flere pakker, potensielt også de som blir forsinket gjennom Internett, og dermed minske effekten av pakketap. Dette vil derimot resultere i høyere ende-til-ende forsinkelse fordi bufferen må vente lenger før lyden blir avspilt. Dante har i skrivende stund ingen mulighet for å øke bufferen høyere enn de valgene man har i Dante Controller.

Flere AoIP-løsninger bruker teknikker for å redusere effekten av pakketap. En mye brukt metode er **Forward Error Correction (FEC)**, som innebærer å sende redundant informasjon med pakkene som kan gjenskape tapte data. Det betyr at i tillegg til den originale dataen i en pakke, legges det til litt data om den forrige pakken som ble sendt. Ved pakketap kan mottakeren gjenskape den tapte pakken lokalt, ut fra tilleggsdataen som kommer med neste pakke, og avsenderen trenger ikke sende en ny pakke. En implementering av dette kan redusere opp til 10% av pakketap, men krever høyere båndbredde fordi hver pakke får mer data [14].

Dersom det ene tunnelendepunktet er kjent og forblir den samme over lengre tid kan man sette opp BPI-R2 til å automatisk koble seg opp til den under oppstart. Dette vil forenkle konfigurasjonen og betyr at man ikke trenger å sette tunnelen manuelt. Her kan BPI-R2 også settes opp til å gjennomføre samme *Traceroute*-script som ble brukt under testingen i kapittel 4.1.1 automatisk, for å "slå hull" på NAT, slik at man slipper å sette opp *port-*

*forwarding* i ruteren.

VXLAN-tunnelering støtter multicast. Det vil si at en avsender kan ha flere mottakere. Under dette prosjektet ble det bare sendt data fra én avsender til én mottaker, men multicast åpner opp for potensielle muligheter med sammenkobling av flere studioer samtidig. For at multicast-trafikk skal sendes over Internett krever det at nettleverandøren har rutere som støtter multicast, og det er det flere som gjør gjennom *Protocol Independent Multicast (PIM)* protokollen [42]. Dette åpner for å sette opp multicast VXLAN-tunneler mellom flere tunnelendepunkter, og dermed gjennomføre produksjoner med flere studioer samtidig.

# Bibliografi


[1] Dani Deahl. *How this Grammy-winning producer turns his laptop into a studio*. The Verge. Library Catalog: www.theverge.com. 28. sep. 2018. URL: https://www.theverge.com/2018/9/28/17874576/music-production-laptop-studio-producer (sjekket 07.03.2020).

[2] David Pierce. «The Guy Who Produced Kendrick Lamar's Best New Track Did It All on His iPhone». I: *Wired* (14. apr. 2017). ISSN: 1059-1028. URL: https://www.wired.com/2017/04/steve-lacy-iphone-producer/ (sjekket 07.03.2020).

[3] Josh Eells. *Billie Eilish and the Triumph of the Weird*. Rolling Stone. Library Catalog: www.rollingstone.com. 31. jul. 2019. URL: https://www.rollingstone.com/music/music-features/billie-eilish-cover-story-triumph-weird-863603/ (sjekket 08.03.2020).

[4] *SVT - groundbreaking Remote Production with Net Insight solution*. Net Insight. Library Catalog: netinsight.net Section: Case Studies. 2. okt. 2019. URL: https://netinsight.net/resource-center/case-studies/svt-remote-production/ (sjekket 03.06.2020).

[5] Marius Jørgenrud. *86 prosent har tilgang til bredbånd med høy hastighet*. Digi.no. Library Catalog: www.digi.no. 27. sep. 2019. URL: https://www.digi.no/artikler/86-prosent-har-tilgang-til-bredband-med-hoy-hastighet/475071 (sjekket 04.06.2020).

[6] Alan P. Kefauver og David Patschke. *Fundamentals of digital audio*. New ed. The computer music and digital audio series v. 22. OCLC: ocn122261679. Middleton, Wis: A-R Editions, Inc, 2007. 190 s. ISBN: 978-0-89579-611-0.

[7] Lee Minich. «A Crash Course In CobraNet». I: *Systems Contractor News* 14.8 (aug. 2007), s. 1. ISSN: 10784993. URL: https://search.proquest.com/docview/201372271?accountid=12870.

[8] D. Y. B. csapat. *East Connection Music Recording CO. | Budapest Art Orchestra - recording session, scoring session, music recording services*. Library Catalog: eastconnection.org. URL: http://eastconnection.org/ (sjekket 03.06.2020).

[9] *Source Elements - Features: AutoRestore and Replace*. URL: https://source-elements.com/features/restore-replace (sjekket 14.06.2020).

[10] *Source Elements - Source-Connect: Features and pricing*. URL: https://source-elements.com/products/source-connect/versions (sjekket 13.06.2020).

[11] Juan-Pablo Cáceres og Chris Chafe. «JackTrip: Under the Hood of an Engine for Network Audio». I: *Journal of New Music Research* 39.3 (sep. 2010), s. 183–187. ISSN: 0929-8215, 1744-5027. DOI: 10.1080/09298215.2010.481361. URL: http://www.tandfonline.com/doi/abs/10.1080/09298215.2010.481361 (sjekket 03.06.2020).

[12] *Dante AVIO Networking Adapters | Audinate*. URL: https://www.audinate.com/products/devices/dante-avio (sjekket 09.06.2020).

[13] Audinate Inc. *Death of Analogue and the Rise of Audio Networking*. Library Catalog: go.audinate.com. URL: http://go.audinate.com/resources/assets/death-of-analog-rise-of-audio-networking-f (sjekket 14.06.2020).

[14] James F. Kurose og Keith W. Ross. *Computer networking: a top-down approach*. Seventh edition, global edition. OCLC: 966602784. Boston Columbus Indianapolis Amsterdam Cape Town: Pearson Education, 2017. 852 s. ISBN: 978-1-292-15359-9.

[15] George Varghese. «An interview with Leonard Kleinrock». I: *Communications of the ACM* 62.11 (24. okt. 2019), s. 31–36. ISSN: 00010782. DOI: 10.1145/3363183. URL: http://dl.acm.org/citation.cfm?doid=3368886.3363183 (sjekket 18.03.2020).

[16] Douglas Comer. *Internetworking with TCP/IP. Volume one*. Pearson new international edition, sixth edition. Always learning. OCLC: 959923021. Harlow: Pearson, 2014. 626 s. ISBN: 978-1-292-04081-3.

[17] J. Postel. *Internet Protocol*. RFC 791. RFC Editor, sep. 1981. DOI: 10.17487/RFC0791. URL: http://www.rfc-editor.org/rfc/rfc791.txt.

[18] J. Postel. *Transmission Control Protocol*. RFC 793. RFC Editor, sep. 1981. DOI: 10.17487/RFC0793. URL: http://www.rfc-editor.org/rfc/rfc793.txt.

[19] J. Postel. *User Datagram Protocol*. RFC 768. RFC Editor, aug. 1980. DOI: 10.17487/RFC0768. URL: http://www.rfc-editor.org/rfc/rfc768.txt.

[20] «802.3: IEEE Standard for Ethernet». I: *IEEE Std 802.3-2015 (Revision of IEEE Std 802.3-2012)* (mar. 2016). Conference Name: IEEE Std 802.3-2015 (Revision of IEEE Std 802.3-2012), s. 1–4017. DOI: 10.1109/IEEESTD.2016.7428776.

[21] R. Droms. *Dynamic Host Configuration Protocol*. RFC 2131. RFC Editor, mar. 1997. DOI: 10.17487/RFC2131. URL: http://www.rfc-editor.org/rfc/rfc2131.txt.

[22] Stuart Cheshire og Daniel H. Steinberg. *Zero configuration networking: the definitive guide*. 1st ed. OCLC: ocm64580877. Beijing ; Sebastopol, CA: O'Reilly, 2006. 226 s. ISBN: 978-0-596-10100-8.

[23] *Free Pool of IPv4 Address Space Depleted | The Number Resource Organization*. URL: https://www.nro.net/ipv4-free-pool-depleted (sjekket 05.06.2020).

[24] M. Mahalingam mfl. *Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks*. RFC 7348. RFC Editor, aug. 2014. DOI: 10.17487/RFC7348. URL: http://www.rfc-editor.org/rfc/rfc7348.txt.

[25] Silvia Hagen. *IPv6 essentials*. 3rd edition. Sebastopol, CA: O'Reilly Media, 2014. 389 s. ISBN: 978-1-4493-1921-2.

[26] Steve Church og Skip Pizzi. *Audio over IP: building pro AoIP systems with Livewire*. OCLC: ocn311788949. Burlington, MA: Focal Press, 2010. 259 s. ISBN: 978-0-240-81244-1.

[27] *Troubleshooting and Debugging VoIP Call Basics*. Cisco. Library Catalog: www.cisco.com. URL: https://www.cisco.com/c/en/us/support/docs/voice/h323/14081-voip-debugcalls.html (sjekket 31.03.2020).

[28] Chris Chafe mfl. «Effect of Time Delay on Ensemble Accuracy». I: *International Symposium on Musical Acoustics* (2004), s. 5.

[29] «How To Choose Networked Audio Solutions». I: *A V Technology* 10.4 (2017), s. 38–39.

[30] H. Schulzrinne mfl. *RTP: A Transport Protocol for Real-Time Applications*. RFC 3550. RFC Editor, jul. 2003. DOI: 10.17487/RFC3550. URL: http://www.rfc-editor.org/rfc/rfc3550.txt.

[31] IEEE Standard 1588-2019. *IEEE Approved Draft Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*. 2019.

[32] IEEE Instrumentation {and} Measurement Society og International Elektrotechnische Kommission, red. *Precision clock synchronization protocol for networked measurement and control systems: IEC IEEE international standard*. Edition 2.0, 2009-02. IEC international standard / International Electrotechnical Commission 61588. OCLC: 951456715. Geneva: IEC Central Office, 2009. 271 s. ISBN: 978-0-7381-5401-5 978-2-8318-1026-3.

[33] *Aruba 2930F Switch Series*. Hewlett Packard Enterprise Product documentation. Library Catalog: h20195.www2.hpe.com. URL: https://h20195.www2.hpe.com/v2/getdocument.aspx?docname=c05052929&doctype=quickspecs&doclang=en_us&searchquery=&cc=uk&lc=en (sjekket 14.06.2020).

[34] *HP Aruba 2930F 24G 4SFP+ (JL253A) (Switcher)*. Prisjakt. Library Catalog: www.prisjakt.no. URL: https://www.prisjakt.no/product.php?p=4063085 (sjekket 14.06.2020).

[35] *DAD AX32 | The Key to Golden Workflows*. URL: https://www.digitalaudio.dk/PRODUCTS/AX32-Audio-Router---AD-DA-DD-Converter (sjekket 14.06.2020).

[36] *AES Standard » AES10-2008 (r2019): AES Recommended Practice for Digital Audio Engineering - Serial Multichannel Audio Digital Interface (MADI)*. Library Catalog: www.aes.org. URL: http://www.aes.org/publications/standards/search.cfm?docID=17 (sjekket 14.06.2020).

[37] *REDNET D64R | Focusrite Audio Engineering Ltd.* URL: https://pro.focusrite.com/category/audiooverip/item/rednet-d64r (sjekket 14.06.2020).

[38] *RED 4PRE | Focusrite Audio Engineering Ltd.* URL: https://pro.focusrite.com/category/audio-interfaces/item/red-4pre (sjekket 14.06.2020).

[39] *Can Dante operate over a Wi-Fi network? | Audinate | FAQs*. Library Catalog: www.audinate.com. URL: https://www.audinate.com/learning/faqs/can-dante-operate-over-a-wi-fi-network (sjekket 08.06.2020).

[40] *Which audio applications does Dante Virtual Soundcard support? | Audinate | FAQs*. Library Catalog: www.audinate.com. URL: https://www.audinate.com/learning/faqs/which-audio-applications-does-dante-virtual-soundcard-support (sjekket 09.06.2020).

[41] *ComputerSalg.no : BananaPi R2 Quad Core 2GB DDR3*. Library Catalog: www.computersalg.no. URL: https://www.computersalg.no/i/3696261/bananapi-r2-quad-core-2gb-ddr3 (sjekket 13.06.2020).

[42] B. Fenner mfl. *Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)*. RFC 7761. RFC Editor, mar. 2016. DOI: 10.17487/RFC7761. URL: http://www.rfc-editor.org/rfc/rfc7761.txt.

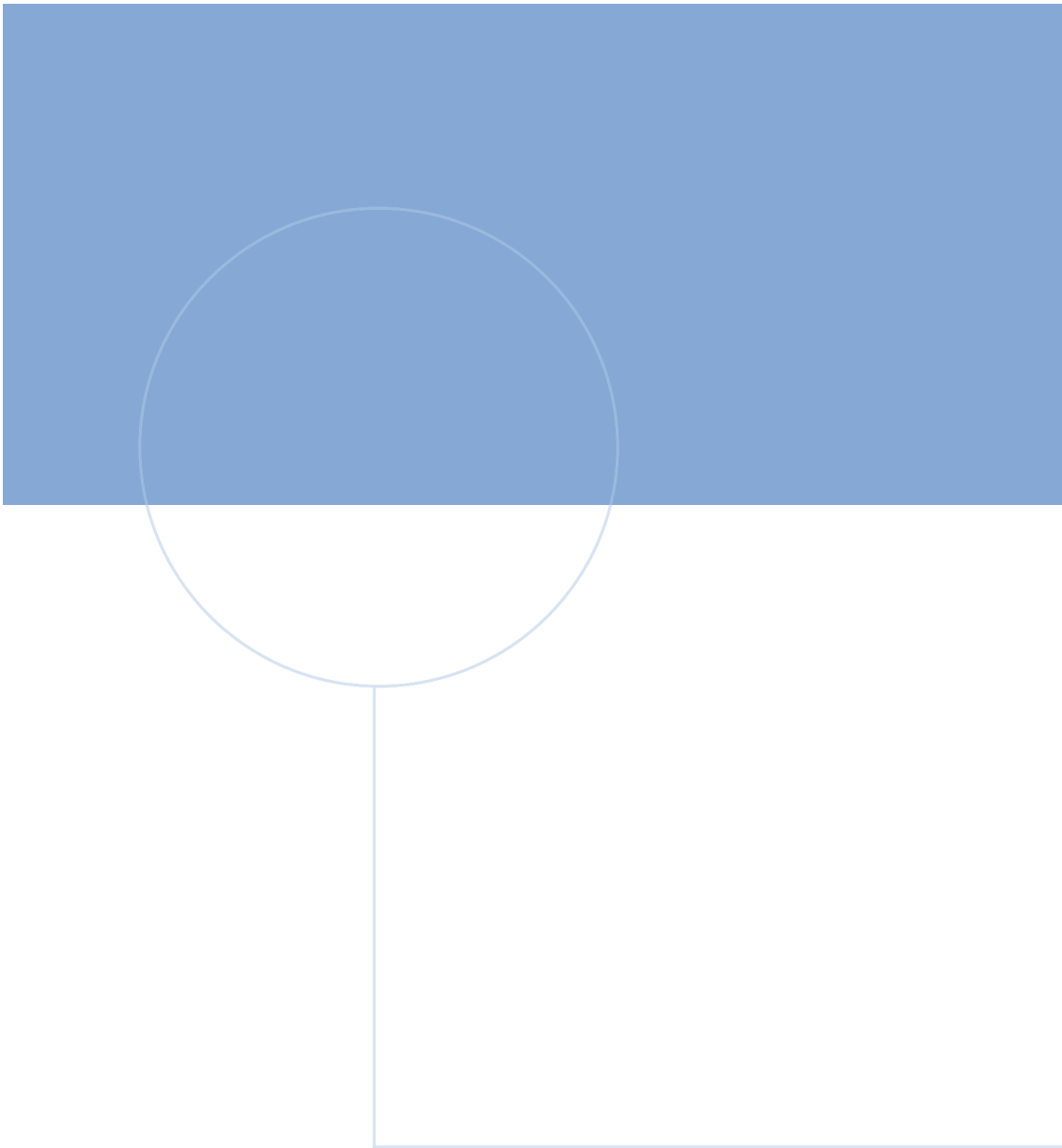
# Tillegg A

# Script

```
#! /bin/bash
#
# Create and setup vxlan tunnel
#

USAGE="Usage: ‘basename $0‘ [-m mcaddr | -r hostaddr | -p port |-h] tunnel-device [bridge-device]\n
  -h              help message\n
  -m mcaddr       multicast address\n
  -r hostadd      host address of remote tunnel endpoint\n
  -p port         port of multicast address og remote tunnel endpoint\n
  -i ip address   Ip address of tunnel inside\n
  -X ip address   Ip address of tunnel inside\n
  -P \"minport maxport\"          Min and max port in source portt range\n
  -d              delete tunnel\n
  tunnel-device   device name of local tunnel endpoint\n
  bridge-device   name of device to bridge with tunnel\n
  "

#declare -A PARAM # Array of parameters
#PARAM=([w]=640 [h]=480 [fr]=30/1)

MCADDR="233.0.224.4"
RHOST=""
RPORT="4789"
VXLAN="vxlan1"
VXLANIP="10.100.100.120"
VXBRIDGE="vxbr1"
SRCPORT="4789 4789"

ACTION="add"

# Parse options
```

```
33 while getopts "m:r:p:i:hd" opt; do
34     case $opt in
35   m)
36       MCADDR=$OPTARG;
37       ;;
38   r)
39       RHOST=$OPTARG;
40       ;;
41   p)
42       RPORT=$OPTARG;
43       ;;
44   P)
45       SRCPORT=$OPTARG;
46       ;;
47   h)
48       echo -e $USAGE
49       exit 0
50       ;;
51   d)
52       ACTION="del"
53       ;;
54   i)
55       VXLANIP=$OPTARG;
56       ;;
57   \?)
58       echo "Invalid option: -$OPTARG" >&2
59       echo -e $USAGE
60       exit 1
61       ;;
62     esac
63 done
64 shift $(($OPTIND - 1))  # (Shift away parsed arguments)
65
66 # root user required
67 if [ `whoami` != 'root' ]
68 then
69     echo "Root access required. Run 'sudo `basename $0` $@'."
70     exit 1;
71 fi
72
73 TDEV="$1"   # Tunnel device
74 BDEV="eno1"   # Bridge device
75
76 # Use multicast as default if no remote host is given
77 ADDR="group $MCADDR ttl 15"
78 if [ "$RHOST" ]
79 then
80     ADDR="remote $RHOST"
81 fi
82
83 case $ACTION in
84     "add")
85   if [ -z "$TDEV" ]
86   then
```

```
87        # No device given. Show status
88        ip -h link show $VXLAN
89        brctl show $VXBRIDGE
90        #ip addr show $VXLAN
91    else
92        # Create tunnel endpoint
93        ip link set $VXLAN down 2> /dev/null
94        ip link del $VXLAN 2> /dev/null
95        ip link add $VXLAN type vxlan id 1 $ADDR dstport $RPORT
      srcport $SRCPORT dev $TDEV
96        ip link set $VXLAN up
97        if [ "$BDEV" ]
98        then
99      # Create bridge
100     ifconfig $VXBRIDGE down 2> /dev/null
101     brctl delbr $VXBRIDGE 2> /dev/null
102     brctl addbr $VXBRIDGE
103     brctl addif $VXBRIDGE $VXLAN
104     brctl addif $VXBRIDGE $BDEV
105     ifconfig $VXBRIDGE up
106     ip addr add $VXLANIP dev $VXBRIDGE
107       else
108     ip addr add $VXLANIP dev $VXLAN
109       fi
110   fi
111   ;;
112     "del")
113   # Remove tunnel endpoint
114   ifconfig $VXBRIDGE down 2> /dev/null
115   brctl delbr $VXBRIDGE 2> /dev/null
116   ip link set $VXLAN down 2> /dev/null
117   ip link del $VXLAN
118   ;;
119 esac
120
121
```



NTNU

Kunnskap for en bedre verden