Tampereen ammattikorkeakoulu, amk-tutkinto
Tietotekniikan koulutusohjelma
Tietoliikennetekniikka
Mikko Järvinen

Tutkintotyö

Mikko Järvinen

Wlan mesh-verkko ja sen service discovery

Tutkintotyö, joka on jätetty opinnäytteenä tarkastettavaksi
insinöörin tutkintoa varten Tampereella 05.10.2009

Työn ohjaaja	Lehtori Ari Rantala
Työn tilaaja	Nokia Research Center

Tampere 2009

Tietotekniikka
Tietoliikennetekniikka

Tutkintotyön tekijä	Mikko Järvinen
Tutkintotyön nimi	Wlan mesh-verkko ja sen service discovery
Päivämäärä	28.09.2009
Sivumäärä	38 sivua.
Hakusanat	WLAN, MESH, SERVICE, DISCOVERY
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Tietoliikennetekniikka
Työn teettäjä	Nokia Research Center
Työn ohjaaja	Lehtori Ari Rantala

Työssä käsitellään wlan mesh-verkon service discoverya ja tutustutaan sen eri toteuttamismahdollisuuksiin. Tutkimuksen lisäksi toteutettiin muutama palvelu tarkoitukseen parhaiten soveltuvalla tavalla.

Tutkimukseen käytettiin suurilta osin erilaisia teknisiä luonnosdokumentteja, manuaaleja ja oltiin yhteydessä saman aiheen parissa työskentelevien kehittäjien kanssa foorumien kautta.

Tuloksena oli toimiva wlan mesh-verkko, jota rajoittivat ainoastaan ajurit ja laitekanta. Lisäksi projektin puitteissa tehtiin verkon käyttöönottoa ja service discovery ominaisuuksia helpottava ohjelmisto. Unicast:na service discovery ominaisuudet toimivat, mutta johtuen usb-porttien hitaudesta multicastina video streamit osoittivat käytettävissä olevan kaistan riittämättömyyden.

Näiden tulosten avulla voidaan suorittaa jatkokehitystä verkon potentiaalia rajoittavien tekijöiden suhteen.

Author Mikko Järvinen
Engineering thesis WLAN mesh network service discovery

Date 28th of september, 2009
Number of pages 38 pages.
Keywords WLAN, MESH, SERVICE, DISCOVERY
Degree programme Computer systems engineering
Specialization Telecommunications engineering

Commission company Nokia Research Center

Thesis supervisor Senior Lecturer Ari Rantala

The work deals with wlan mesh networks service discovery and it's various implementation methods. Addition to this research we also implemented some services with the chosen method.

Research material mainly consisted of technical drafts, manuals and communication with other developers through internet forums. As a resul we got a working mesh-network which was only restricted by drivers and availability of suitable devices. A software was made to more easily set up the mesh network and to make use of the service discovery features. In unicast the features worked fine but in multicast the videostream started to freeze up. This was mainly because of the slow usb ports.

With these results further development can be made to overcome restrictive qualities.

ALKUSANAT

Haluan kiittää Demolan, Nokia Research Centerin ja TAMK:n ihmisiä tuesta.

Työ on tehty Demola projektissa Nokia Research Center:lle.

Työ on julkinen.

Tampereella 05.10.2009

Mikko Järvinen

Sisällys

Lyhenteet ja merkit

API	Application programming interface. Rajapinta, jossa määritellään miten sovellus voi kommunikoida ohjelmien ja käyttöjärjestelmien kanssa.
Avahi	Ilmainen zeroconf toteutus linuxille.
DAAP	Digital Audio Access Protocol. Applen luoma protokolla media-tiedostojen jakamiseen verkossa.
HWMP	Hybrid Wireless Mesh Protocol. Oletus reititys protokolla 802.11s:ssä.
IEEE	Institute of Electrical and Electronics Engineers. Voittoa tavoittelematon järjestö, joka edistää sähköön liittyvän teknologian kehitystä.
IETF	Internet Engineering Task Force. Järjestö, joka kehittää ja edistää internet tekniikoita.
Iw	Linuxin langattomien verkkolaitteiden hallintaohjelma.
Kernel	Käyttöjärjestelmän ydin. Joka määrittelee käyttöjärjestelmän toiminnan.
MAC	Media Access Control. IEEE 802-verkoissa tämä osa varaa verkon ja hoitaa liikennenöintiä.
Man-sivut	Linux-käyttöjärjestelmien manuaalisivut.

Mesh	Wlan-standardiin pohjautuva uusi verkkoratkaisu.
PTR	Osoitin tieto, jota hyödynnetään DNS-SD:ssä, kun halutaan vain tieto, mitä palveluja verkossa on.
Service Discovery	Protokollat, jotka mahdollistavat veerkon laitteiden ja palveluiden automaattisen käyttöönoton.
Skripti	Lyhyt tulkattava tietokone ohjelma.
SLP	Service Location Protocol.
SRV	DNS-SD:n käyttämä tieto, jossa kerrotaan mistä ja miten palvelun löytää.
SSDP	Simple Service Discovery Protocol.
TXT	DNS-SD tieto, joka kertoo palvelun yksityiskohtia.
Wlan	Wireless Local Area Network.
Zeroconf	Tekniikka, jolla saadaan aikaiseksi toimiva IP-verkko ilman manuaalista konfigurointia.

1. Johdanto

Standardia ei ollut olemassa, käytettävä laitekanta oli laskettavissa yhden käden sormilla ja käyttöjärjestelmäksi oli vain yksi vaihtoehto. Tästä huolimatta työhön käytiin innolla ja hitaan alun jälkeen saatiin tuloksiakin. Työni esittelee palveluiden toteuttamista mesh-verkossa, niiden toimintaa ja tarjoaa myös muutaman toteuttamamme palvelun.

Työssä käytetty verkkoratkaisu ei tarvinnut lainkaan perinteisesti verkkoon kuuluvia elementtejä ja mikä parasta se oli ilmainen. Verkon liikenne kulki päätteiden välillä suoraan. Palveluiden puolesta tämä ei aiheuttanut muita ongelmia, kuin verkon kapasiteetin loppumisen kesken. Tähän omalta osaltaan vaikutti myös ajureiden/standardin määrittelyt.

Toteutetut palvelut toimivat, vaikkakin jossain tapauksissa hitaasti edellä mainitusta syyystä johtuen. Automaattiset IP-osoitteet toimivat niin ikään ilmaisella ohjelmalla. Palveluiden toteuttamiseen ja niiden toimintaan perehtyessä tuli mieleen vain yksi ajatus: Miksi tätä ei ole tehty jo aikaisemmin, sillä tämähän on käyttökelpoista ja näppärää.

2. Mikä on mesh?

Mesh-verkossa voidaan siirtää dataa siinä, missä muissakin verkoissa. Siinä on kuitenkin mahdollista jatkuviin reitin muutoksiin ja hyppyihin, jotta data pääsee päätepisteeseensä. Mesh-verkon ominaisuuksiin kuuluu myös se, että verkon päätteet voivat keskenään muodostaa täysin toimivan verkon, joka yleensä ei ole liikkuva. /1/

Mesh-verkko ratkaisut ovat kuitenkin liikkumassa langattomuuden suuntaan osittain syystä, että ne alunperin kehiteltiin sotilaallisin tarkoituksiin mutta radiolaitteiden kehittyessä ja halventuessa ovat monet muut langattomat mesh-verkko ratkaisut tulleet esiin. /1/

2.2 IEEE 802.11s

Työssämme käytetty langaton mesh-ratkaisu on lisäys tunnettuun 802.11-tekniikkaan. Siinä lisätään 802.11 MAC:iin arkkitehtuuri ja protokollat, jotka tukevat broadcastia, multicastia ja unicast toimitusta. /2/

Laitteita 802.11s-verkossa kutsutaan mesh station:ksi. Asemat muodostovat linkkejä toistensa välille, joiden päälle voidaan muodostaa reittejä käytössä olevalla reititysprotokollalla. Oletuksena 802.11s:ssä on *Hybrid Wireless Mesh Protocol* (HWMP), jos mitään muuta protokollaa ei ole määritelty on *HWMP* oletuksena käytössä oleva protokolla. /2/

Mesh-asemat eivät siis tarvitse välilleen erikseen reitittimiä tai muita verkon perinteisiä laitteita. Verkkoon on kuitenkin mahdollisuus lisätä mesh portaaleja, joiden avulla voidaan yhdistää eri mesh-verkkoja tai muita verkkoja mesh-verkon lävitse. Tällä menetelmällä on mahdollista saada mesh-verkkoon internet yhteys. /3/

3. Mesh-verkon pystytys ja testiverkkomme

Työ alkoi sopivien laitteiden etsinnällä ja projektin käyttöön sopivan käyttöjärjestelmän valinnalla.

3.1 Laitteet

Työssä käytettäväksi wlan-sovittimeksi valittiin A-link:n WL54USB-sovitin tarkoitukseen sopivan piirisarjan takia. Aluksi testatun D-linkin piirisarjan vaihtui sellaiseen, jolla tarvittavat ajurit eivät toimineet.

Testaamisen helpottamiseksi tietokoneiksi valittiin kannettavat tietokoneet.

Projektimme tietokoneet olivat seuraavanlaisia:

Kone	Prosessori	Muisti
Acer Aspire 5630	Intel Core2 T5200 1.6 GHz	1 Gt
IBM T41	Intel Pentium 1.6 GHz	1 Gt
IBM T41	Intel Pentium 1.6 GHz	1 Gt
IBM T41	Intel Pentium 1.6 GHz	1 Gt

Taulukko 1: Projektissa käytetyt tietokoneet.

Kaikki koneet soveltuivat hyvin projektin tarkoituksiin, vaikkakin yksi ibm:n koneista aiheutti ongelmia ilmeisesti viallisen usb-portin takia. Acer osoitti etunsa video- ja audio-testeissä, sillä sen äänikortti toimi myös käytetyssä testikernelissä.

3.2 Käyttöjärjestelmän valinta ja sen asennus testikäyttöön

Käyttöjärjestelmän suhteen ei ollut juurikaan vaihtoehtoja, kun kaikki tarvittavat ajurit on tehty linux/unix-järjestelmille. Valintaa rajoitti tai helpotti myös tärkeimpien ohjeiden löytyminen käytetylle käyttöjärjestelmälle. Päädyimme siis lähinnä olosuhteiden pakosta ubuntu-nimiseen linux-versioon.

Perus asennus oli suoraviivainen prosessi eikä vaatinut sen kummempaa tietämystä. Paitsi kovalevyjen osioinnissa, sillä halusimme säilyttää windows-käyttöjärjestelmän koneilla varmuuden vuoksi.
Ubuntun saattaminen testikäyttöön vaati huomattavasti enemmän aikaa ja kärsivällisyyttä, sillä monessa kohtaa prosessi kesti reilusti toista tuntia. Testikäyttöön saattaminen tapahtui kolmessa vaiheessa: esivaatimukset, kernelin konfiguroiminen ja kernelin kääntäminen.

3.2.1 Esivaatimukset

Aluksi piti asentaa paketit joiden avulla kernel-pakettien teko ja kääntäminen onnistuu. Tämä onnistui helposti alla olevalla komennolla:

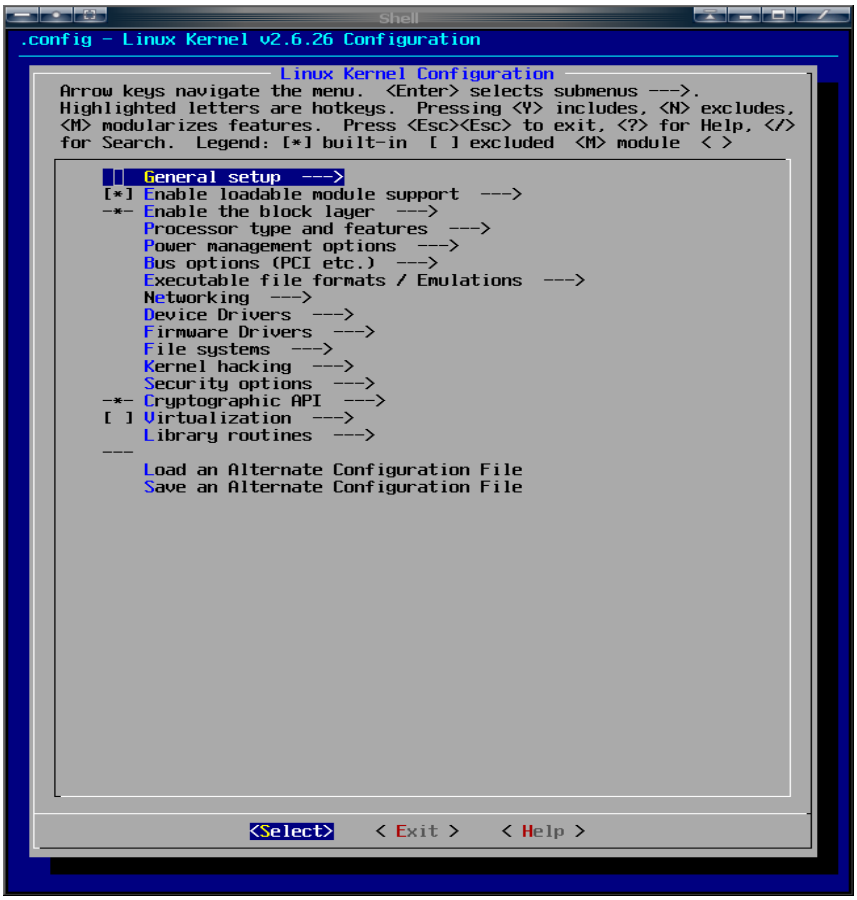
sudo apt-get install *-y fakeroot build-essential git-core kernel-package* /4/

Lihavoitu osa on peruskomento, jolla asennetaan paketteja komentoriviltä ubuntu-linuxissa. Valitsimella ”y” komento vastaa automaattisesti kyllä kaikkiin kyselyihin jotka tulevat asennuksen aikana eikä tulosta niitä ruudulle. Valitsimen jälkeen tulevat asennettavien pakettien nimet välilyönneillä eroteltuna.

Kernelin konfigurointia varten asennettiin ncurses-paketit, joita tarvitaan menuconfig:n käyttöön.

sudo apt-get install libncurses5 ncurses-dev /4/

Näin pystytään käyttämään menuconfig-ohjelmaa, jolla kernelin konfigurointi on helppoa valikko-rakenteen vuoksi. Edellyttäen kuitenkin tietämystä siitä, mitä on kerneliin asentamassa.



Kuva 1. Menuconfig-ohjelma.

Esivalmistelujen lopuksi kloonattiin tarvittava testikerneli internet-lähteestä git-komennon avulla.

git clone git://git.kernel.org/pub/scm/linux/kernel/git/linville/wireless-testing.git /4/

Tässä kohtaa pitää olla tarkkana, sillä komento kloonaa kernelin osoitetusta lähteestä siihen hakemistoon, jossa sattuu olemaan.

3.2.2 Kernelin konfigurointi

Kernelin konfigurointi alkoi kopioimalla nykyisen kernelin konfiguraatio testi-kernelin hakemistoon normaalilla kopio-komennolla ”cp” ja käyttämällä sen yhteydessä hyödyksi ”uname”-komentoa ja sen valitsinta ”r”. Valitsimella ”r” saadaan vaivattomasti käytössä olevan kernelin versio ja vältetään kirjoitusvirheet.

cp /boot/config-`uname -r` .config /4/

Kopioitu konfiguraatio liitetään sitten testi-kerneliin. Näin saadaan kätevästi kaikki asetukset joihin ei ole tarvetta tehdä muutoksia.

make oldconfig /4/

Konfiguroinnin lopuksi lisättiin muutama oma asetus menuconfig-ohjelmalla testi-kernelin asetuksiin. Käytettävistä laitteista huolimatta pitää geneerisestä IEEE 802.11 verkko pinosta laittaa päälle tuki mesh-verkoille.

Networking -> Wireless -> Generic IEEE 802.11 Networking Stack (mac80211)
Enable mac80211 mesh networking (pre-802.11s) support /4/

Lisäksi tulee laittaa päälle tuki käytettävien laitteiden ajureille.

Device Drivers -> Network Device Support -> Wireless LAN -> Käytössä oleva laite. /4/

3.2.3 kernelin kääntäminen

Kernel-pakettien tekemiseen ja kääntämiseen käytetään ”fakeroot” ja ”make-kpkg”-komentoja. *Fakeroot*-komennolla make-kpkg uskoo saaneensa pääkäyttäjän oikeudet, vaikka todellisuudessa niitä ei olekkaan. Komennolla *make-kpkg* luodaan itse kernel ja sen header-tiedosto. Valitsimella *initrd* otetaan kernelissä käyttöön *initial ramdisk*, joka on väliaikainen tiedostojärjestelmä ennen kernelin lataamista.

fakeroot make-kpkg --initrd kernel_image kernel_headers /4/

Edellisellä komennolla luodun kernelin tiedostojen asentaminen tapahtui seuraavasti.

sudo dpkg -i linux-.deb* /4/

Tällä komennolla saatiin tiedostot kopioitua /boot-hakemistoon sekä merkintä */boot/grub/menu.lst* tiedostoon, jotta testi-kerneli näkyisi käynnistys valikoissa. Menuconfig-ohjelmalla tehdyt muutokset (modulit) asentuvat */lib/modules* hakemistoon.

3.3 Mesh-verkon pystytys

Mesh-verkon saattaminen käyttöön vaatii useita komentoja ja ne pitää ajaa jokaisella koneella. Testaamisen ja muun käytön nopeuttamiseksi työstimme skriptin, joka ajaa nämä komennot kerralla.

Osa skriptin komennoista vaatii iw-ohjelman asentamista, jolla voidaan hallinnoida verkon laitteita.

3.3.1 Meshin käynnistys skripti

Pysäytetään nm-applet eli NetworkManager-ohjelma täysin. *Killall* komento pysäyttää kaikki määritellyn ohjelman prosessit.

killall NetworkManager

Luodaan mesh interface mesh_id:llä "moi". Tämän täytyy olla sama muissakin mesh-verkon koneissa. Valitsimella ”*dev*” osoitetaan laite, jolle lisätään ”*interface*”. Valitsimella ”*add*” lisätään tässä komennossa mesh-verkon mp eli meshpoint edellä mainitulla mesh id:llä.

iw dev wlan1 interface add mesh type mp mesh_id moi /3/

Tarkistetaan meshin tila. Otetaan verkonlaitteista kaikki tiedot mutta rajataan se vain niihin tuloksiin, joissa esiintyy sana mesh.

ifconfig -a | grep mesh /3/

Otetaan aiemmin luotu mesh *interface* käyttöön, kun sen tila on varmistettu edellisellä komennolla.

ifconfig mesh up /3/

Asetetaan nopeus 54M, sillä oletuksena käytössä on alhaisin mahdollinen wlan-nopeus eikä se riitä esimerkiksi video streamin vastaanottoon.

iwconfig mesh rate 54M /5/

Otetaan käyttöön ip-osoite avahin autoipd toiminnolla. Komennon nimen jälkeen annetaan interface:n nimi jolla halutaan ip-osoite. Valitsimella ”daemon” avahi-autoipd menee daemon moodiin eikä näin ollen varaa terminaali-ikkunaa josta sen ajettu.

avahi-autoipd mesh –daemon /6/

3.4 Verkon testit

Valmiilla verkolla saatoimme suorittaa sarjan testejä, joilla pystyi testaamaan laitteiden tehokkuutta sekä mesh-verkon toimivuutta.

Testi 1:
Normaali etäisyyteen pohjautuva testi kahdella koneella demolan tiloissa finlayssonilla.

Tulokset:
Toimi hyvin. Kulkuajat eivät muodostuneet pitkiksi, eikä paketteja hukkunut.

Testi 2:
Etäisyyteen pohjautuva testi, jossa reitillä oli ovia sekä rappukäytävä. Ensin kahdella koneella ja uudestaan kolmella.

Tulokset:
Ympäristö(ovet, rappukäytävä) esti yhteyden muodostumisen kahden koneen välillä. Yhteys saatiin aikaiseksi lisämäällä yksi kone puoliväliin rappukäytävää ”väliasemaksi”. Aluksi pingaamalla kulkuajaksi muodostui n. 1000 ms mutta ne normalisoituivat hetkenkuluttua.

Testi 3:
Tiedonsiirtonopeustesti 3 Mt:n tiedostolla. Lyhyt etäisyys.

Tulokset:
Saavutimme 100 Kt / s nopeuden eli 1 Mt nopeus saavutettiin. Testin aikaisilla asetuksilla tämä oli suurin mahdollinen nopeus.

Testi 4:

Verkkopelin yhteystesti. Testi pelinä käytettiin ubuntusta löytyvää Freecol-peliä.

Tulokset:

Yhdistäminen toimi, kuten normaalissakin verkossakin ja peliä pystyi pelaamaan.

Testi 5:

Sarja etäisyyteen ja ympäristöön perustuvia testejä TAMK:ssa.

Tilanne 1:

Ruokala – Kuntokadun aula. Ilman kolmatta konetta välissä yhteys toimi luotettavasti palo-ovelle asti(n. 30 m.). Yhden koneen asettuessa tähän pisteeseen ja kolmannen sijoittuessa kuntokadun aulaan(n. 50 m) saatiin yhteys toimimaan koko välillä.

Tilanne 2:

Ruokala – Teiskontien aula. Kahdella koneella yhteys muodostui luotettavasti aulaan asti(n. 50 m). Yhteys katkesi vasta toisen koneen mentyä noin 15 metrin matkan infokahvilaan asti. Kolmas kone ei auttanut tässä tilanteessa.

Tilanne 3:

Teiskontien aula – 6. kerros. Kahdella koneella saavutettiin yhteys kolmanteen kerrokseen. Yhden koneen jäädessä kolmanteen kerrokseen saatiin yhteys ulotettua kuudenteen kerrokseen asti. Tässä pystyttiin toteamaan varmasti, että liikenne kulki kolmannessa kerroksessa olevan koneen kautta.

4. Service Discovery

Service discovery protokollat mahdollistavat palveluiden ja laitteiden automaattisen käyttöönoton lähiverkossa. /7/

Seuraavassa käsittelen tunnetuimmat service discovery-protokollat sekä Applen Zeroconf-tekniikan, jonka avahi nimistä implementaatiota käytimme työssämme ip-verkon toteutukseen.

4.1 Applen protokolla

Applen protokolla on kaksiosainen. Toinen osa hoitaa normaalia DNS-toimintaa ja toisen tehtävänä on hoitaa service discovery toiminnallisuutta. /8/

4.1.1 Multicast DNS

DNS-toiminnallisuus on toteutettu multicast DNS:nä ja se käyttää samanlaisia *API:ja*, kuin normaali DNS, mutta hieman eri tavalla. Jokainen kone lähiverkossa pitää kirjaa DNS-resursseista. /8/

Kun mDNS järjestelmässä halutaan tietää PC:n ip-osoite lähetetään kyseisen koneen nimi(esim. purkki.local) multicast DNS-osoitteeseen, joka link-local verkoissa on 224.0.0.251. Kaikki isäntä-laitteet kuuntelevat näitä pyyntöjä ja se kone vastaa, jonka nimi löytyy mDNS-pyynnöstä. Link-local osoitteet vastaavat ip-muodossa 169.254/16-sarjan osoitteita. *Multicast*-osoitteita voidaan käyttää linkki- ja internet-kerroksella (OSI-mallin kerrokset 2 ja 3). Lisäksi on mahdollisuus käyttää Ipv6:sta Ipv4:sen sijaan. /8/ /9/

4.1.2 DNS-SD

Myös service discovery-toiminnallisuus perustuu DNS:ään. Palveluiden mainostamiseen käytetään DNS:än PTR-, TXT- ja SRV-tietoja. /8/

Kaksi käyttämistämme ohjelmista, joilla loimme palveluja testiverkkoon käytti SRV:tä palvelunsa mainostamiseen. SRV on kätevä, kun etsitään verkosta tietyn tyyppisiä palveluita. Ohjelmat, joilla selataan palveluita lukevat SRV-tiedoista mistä kyseisen palvelun voi löytää. Alla SRV:n perusrakenne:

_Service._Proto.Name TTL Class SRV Priority Weight Port Target

Service viittaa palvelulle valittuun tai muodostuneeseen nimeen.
Proto kertoo käytetyn protokollan, joka usein on TCP tai UDP.
Name kohtaan tulee se domain, jossa kyseinen SRV-tieto on pätevä.
TTL on normaali DNS:n Time-To-Live kenttä.
Class DNS:n kenttä, joka hyvin usein on IN (Internet).
Priority kertoo kohteen ”tärkeyden”, mitä pienempi arvo sitä tärkeämpi.
Weight painoarvo niille SRV-tiedoille joilla on sama Priority-arvo.
Port Portin numero josta palvelu löytyy.
Target sen koneen perusmuotoinen nimi, joka tarjoaa palvelua. /10/

Muutama esimerkki käyttämämmiemme ohjelmien SRV-tiedoista.

_vlc-http._tcp VLC:n SRV perusmuodossaan.
_daap._tcp Rhythmboxin SRV perusmuodossaan. /11/

TXT-tieto tarjoaa lisätietoja palvelusta. Sen koko maksimissaan voi olla 65535 tavua pitkä. Sen koko määritellään DNS-viestin resurssi tiedon otsikossa. Kuitenkin käytettäessä mDNS:ää pakettien maksimi kooksi tulee 9000 tavua, johon luetaan kaikki tarvittavat otsikko tiedot (IP, UDP ja DNS). Tämä rajoittaa TXT-tiedon koon maksimissaan 8900 tavuun, joka käytännössä on vain joitakin satoja tavuja. /12/

Data TXT-tiedon sisällä koostuu yhdestä tai useammasta merkkijonosta, jotka on sijoitettu muistiin peräkkäin ilman välejä tai täytetavuja. Yksittäinen merkkijono koostuu yhdestä tavusta, jonka jälkeen tulee 0-255 tavua teksti dataa. /12/

DNS-SD käyttää TXT-tietoja omavaltaisten avain – arvo parien välittämiseen. Nämä parit kertovat lisätietoja kyseistä palvelusta. Jokainen pari koodataan omaan merkkijonoonsa muodossa avain=arvo. /12/

TXT-tieto voisi esimerkiksi olla muotoa:

```
-------------------------------------------------------------
| 0x09 | key=value | 0x08 | paper=A4 | 0x07 | passreq |
-------------------------------------------------------------
```

Tämä kertoisi tulostus palvelulle, että sen pitäisi käyttää A4-kokoista paperia ja salasanaa vaaditaan. /12/

Kun verkosta etsitään palveluja ei ensimmäiseksi haeta SRV-tietoa vaan niiden sijaan pyydetään PTR-tiedot (Pointer). Tällä saadaan listaus kaikista tietyssä paikassa olevista palveluista. /12/

4.2 Microsoftin protokolla - SSDP

Microsoftin ja Hewlett-Packardin *SSDP* on UPnP-protokolla, joka perustuu jo vahentuneeseen IETF:än internetluonnokseen. *SSDP* on käytössä Windows XP:ssä sekä useissa verkko-ominaisuuksia sisältävissä laitteissa. /8/

SSDP käyttää unicast- ja multicast-paketteja verkon palveluiden mainostamiseen. IPv4-verkoissa multicast osoitteena toimii 239.255.255.250 ja IPv6 verkkossa se vaihtelee käytössä olevan alueen mukaan. *SSDP*-liikenne kulkee portin 1900 kautta. /13/

Palvelut *SSDP:ssä* tunnistetaan *URI* (Uniform Resource Indentifier) ja *USN* (Unique Service Name) parista. *URI:stä* selviää palvelun tyyppi ja *USN:stä* sen nimi. *USN:llä* siis pystytään erottelemaan saman tyyppiset palvelut toisistaan. Kun palvelusta annetaan sekä *URI* että *USN* palvelu löydökset ja palveluilmoitukset sisältävät myös sijaintitietoja sekä erääntymisajan. /14/

USN:än tarkoituksena on antaa palvelulle nimi johon voi aina viitata, vaikka laitteen oma nimi muuttuisikin. Ilman *USN:ää* sijainnin muuttuessa järjestelmä luulisi, että kyseesä olis aivan uusi palvelu, vaikka tosiassa näin ei ole. /14/

Sijainti tieto kertoo kuinka kyseiseen palveluun tulisi ottaa yhteys. Erääntymisaika puolestaan kertoo kuinka kauan kyseisen palvelun tietoja säilytetään välimuistissa ja kuinka kauan palvelun oletetaan olevan tavoitettavissa. /14/

Palvelu välimuisti voisi esimerkiksi näyttää tältä /14/ :

USN URI	Service Type URI	Expiration	Location
upnp:uuid:k91...	upnp:clockradio	3 days	http://foo.com/cr
uuid:x7z...	ms:wince	1 week	http://msce/win

Poiketen muista SD-protokollista *SSDP:ssä* on erikseen palvelun ilmoitus- ja hakuviestit. Tällä vähennetään verkon liikennettä. Useat muut protokollat ovat jommassakummassa ääripäässä. Ne joko lähettävät pelkkiä haku- tai ilmoitusviestejä verkkoon. /14/

SSDP:ssä ratkaisu on ääripäiden väliltä. Uuden palvelun tullessa verkkoon se lähettää yhden ilmoituksen itsestään ja verkkoon kirjautuessa käyttäjät lähettävät yhden hakuviestin. Lisä viesteille ei pitäisi olla tarvetta, sillä palvelut joiden yhteys verkkoon katkeaa ilmoittavat itsestään. Tämä vähentää verkon liikennettä, mutta tekee *SSDP:stä* monimutkaisemman protokollan. /14/

Muihin protokolliin poikkeuksena on myös aiemmin mainittu erääntymisaika. Tämän ajan kuluttua loppuun käyttäjien ei tarvitse odotella ilmoitusviestiä vaan he voivat lähettää hakuviestin. /14/

Hakuviesteissä pitää olla palveluntyyppi otsikko. Tämä otsikko sisältää yhden *URI:n* ja sitä voidaan käyttää haettavan palvelun rajaamiseen. Vain ne palvelut joiden tyyppi vastaa määrittelyä voivat vastata hakuviestiin. Hakuviestit lähetetään multicastina ja palvelut vastaavat suoraan viestin lähettäneen käyttäjän ip-osoitteeseen/porttiin. /14/

Vastaus hakuviestiin sisältää palvelun sijainnin sekä *USN*- ja palveluntyyppiotsikot. Vastauksen tulisi myös sisältää välimuistin hallintaa vanhetumis otsikon tai ”*max-age*” määritelmän muodossa, jos mitään näistä ei ole määritelty käyttäjän ei tule tallentaa vastauksen tietoja välimuistiinsa. Molempien ollessa määriteltyinä välimuistin hallinta on etusijalla. /14/

Esimerkki hakuviestistä /14/ :

```
M-SEARCH * HTTP/1.1
S: uuid:ijklmnop-7dec-11d0-a765-00a0c91e6bf6 ← Lähde
Host: 239.255.255.250:reservedSSDPport ← multicast-osoite
Man: "ssdp:discover"
ST: ge:fridge ← Palvelun tyyppi.
MX: 3
```

Esimerkki hakuviestin vastauksesta /14/ :

```
HTTP/1.1 200 OK
S: uuid:ijklmnop-7dec-11d0-a765-00a0c91e6bf6
Ext:
Cache-Control: no-cache="Ext", max-age = 5000 ← Välimuistinhallinta
ST: ge:fridge ← Palvelun tyyppi
USN: uuid:abcdefgh-7dec-11d0-a765-00a0c91e6bf6 ← Palvelun USN
AL: <blender:ixl><http://foo/bar> ← Sijainti
```

Ilmoitusviestillä on kaksi tyyppiä. Alive-tyyppisellä viestillä ilmoitetaaan uudesta palvelusta, päivitetään tiedot erääntymisajan umpeutuessa ja sijainnin muuttuessa ilmoitetaan uusi sijainti. Byebye-viestillä kerrotaan palvelun lopettavansa toimintansa. /14/

Alive-tyyppisen ilmoitusviestin rakenne on hakuviestin vastauksen kaltainen /14/ :

```
NOTIFY * HTTP/1.1
Host: 239.255.255.250:reservedSSDPport ← Multicast-osoite
NT: blenderassociation:blender ←  Palvelun tyyppi
NTS: ssdp:alive ←  Viestin tyyppi
USN: someunique:idscheme3 ← USN
AL: <blender:ixl><http://foo/bar> ← Sijainti
Cache-Control: max-age = 7393 ← Välimuistin hallinta
```

Byebye-tyyppisen ilmoitusviestin rakenne /14/ :

```
NOTIFY * HTTP/1.1
Host: 239.255.255.250:reservedSSDPport ← Multicast-osoite
NT: someunique:idscheme3
NTS: ssdp:byebye ← Viestin tyyppi
USN: someunique:idscheme3
```

Näiden viestityyppien lisäksi voidaan käyttää all-tyyppistä viestiä, jolla voidaan hakea verkon kaikki palvelut. /14/

4.3 IETF:n standardi protokolla SLP

Ainoa service discovery protokolla, joka on *IETF:n* esitetyn standardin asteella. Protokollaa eteenpäin ajavat HP, Novell, Sun ja Apple. /8/

SLP:ssä käyttäjiin viitataan ”*User agent*”:na ja palvelut puolestaan ”*Service agent*”:na. Kolmanten osapuolena verkossa toimii ”*Directory agent*”, jonka avulla protokollaan saadaan skaalautuvuutta. /15/

Kahden ensiksi mainitun avulla saadaan jo toimiva ratkaisu aikaan. User agent:it lähettävät verkkoon *multicast*-viestinä *service requestin*, jossa on määritelty minkälaista palvelua ollaan hakemassa. Paluuviestinä se saa unicastina lähetettyjä *Service reply*-viestejä, joissa puolestaan on määritelty mistä haettu palvelu löytyy. /15/

Suuremmissa verkossa käytetään kuitenkin *Directory agent:a*, joka toimii palveluiden välimuistina. *User agent:it* lähettävät samanlaisen viestin, kuin edelläkin, mutta sen kohteena on *Directory agent*. *Service agent* lähettää *Directory agent:lle* palvelustaan rekisteröinti viestin, edellisestä kuitenkin poiketen tämä lähetetään unicast-viestinä. *Directory agent* vastaa *User agent:lle* niinkuin edelläkin ja *Service agent:lle* se lähettää kuittauksen palvelun rekisteröinnistä välimuistiin. Välimuistissa olevat palvelut tulee uudistaa, jotta ne eivät vanhenisi. /15/

Directory agent:it löydetään kahdella eri tavalla. Ensimmäisessä vaihtoehdossa muut agentit tullessaan verkkoon lähettävät multicastina service requestin *Directory agent:lle*. Toisessa tavassa *Directory agent* lähettävät satunnaisin väliajoin viestejä olemassaolostaan. Kummassakin tavassa muut agentit saavat *DA advertisement* viestin. /15/

Service agent:it ja *User agent:it* voidaan konfiguroida käyttämään *Directory agent:a* dynaamisesti *DHCP*:tä käyttä tai staattisilla osoitteilla. Konfigurointi tavasta huolimatta *Service agent:in* pitää verkkoon tullessaan lähettää unicastina service request *Directory agent:lle*. Tähän *Directory agent* vastaa kaikella tarvittavalla tiedolla, jota tarvitaan *Service agent:in* ja *Directory Agent:in* väliseen kommunikointiin. /15/

User agent löytää *Directory agent:it* staattisen konfiguraation, *DHCP:n* avulla tai multicastina lähetettyjen service requestien avulla. /15/

Kaikki multicastit *SLP:ssä* lähetetään *SLP:n multicast* osoitteeseen, joka on 239.255.255.253. Kaikille viesteille on varattu portti 427. Viestien TTL-arvoksi tulee asettaa 255. Pienemmät arvot saattavat rajoittaa palveluiden löytymistä verkossa. /15/

Palveluiden sijainti ja tyyppi kerrotaan url-tyyppisillä osoitteilla. Osoite voi olla erikseen palveluita varten määritellyn mukainen tai minkälainen url-osoite hyvänsä kunhan se on uri-standardin mukainen. Minkä tahansa verkon palvelun sijainnin voi ilmaista url-osoitteella. /15/

Palvelun osoite on yleensä muotoa /15/ :

"service:"<srvtype>"://"<addrspec>

Palvelun tyyppi määritellään kohtaan srvtype, johon ei lueta kaksoispistettä. Addrspec kohtaan tulee palvelun sijainti ja sen perään mahdollinen portti muodossa :porttinumero, jos porttinumeroa ei ole määritelty otetaan käyttöön varattu portti numero (Esim. ftp:llä 21). Itse sijainti www-osoitteen tyylinen. Epästandardien porttinumeroiden käyttö on myös mahdollista, kuten alla olevasta esimerkistä käy ilmi. /15/

service:tftp://bad.glad.org:8080

4.4 Zeroconf

Zeroconf-tekniikalla saa helposti käyttöönsä IP-verkon ilman servereitä ja ilman monimutkaisia asetuksia. Ilman zeroconf-tekniikkaa pitäisi käytössä olla DHCP-serveri ja DNS-palvelu. Automaattisten ip-osoitteiden lisäksi zeroconf-tekniikalla onnistuu myös automaattinen DNS-toiminnallisuus sekä service discoveryn ja zeroconf:n mahdollistamat palvelut.

Service discoveryn ja Zeroconf:n avulla saa monia laitteita ja palveluita käyttöönsä vaivattomasti verkon yli. Tähän tekniikkaan perustuva, vaikka monikaan ei sitä välttämättä tiedosta on tulostimen käyttöönotto verkon yli ilman mitään erillisiä asennus toimenpiteitä. Muita varsin käteviä palveluita ovat musiikki-serveri, videoiden katselu verkon yli ja FTP-tiedonsiirto. Osa näistä kuulostaa vanhalta mutta näiden vanhojenkin sovellusten käyttöönotto ja käyttäminen testiympäristössämme oli huomattavasti vaivattomampaa, kuin esimerkiksi Windows-käyttöjärjestelmässä.

4.4.1 Zeroconf:n käyttö testiverkossamme

Käytimme työssämme Applen protokollaan perustuvaa Zeroconf-implementaatiota nimeltään Avahi, joka on avoimeen lähdekoodiin perustuva ohjelma. Ennenkuin Avahia voitiin käyttää tuli asentaa useampi paketti.

Asennus onnistui helposti komentoriviltäkin, kun tiesi, mitä paketteja piti asentaa. Pakettien nimien perusteella tein huomion, että osa toiminallisuudesta ei sisälly itse avahiin. Tästä johtuen asennus kannattaa hoitaa käyttämänsä linux-version pakettienhallinta-ohjelmaa (Synaptic ubuntussa).

> *sudo apt-get install avahi-daemon avahi-discover avahi-utils libnss-mdns service-discovery-applet mdns-scan* /16/

4.4.2 FTP-palvelu

Ensimmäisen testasimme FTP-palvelun luomista verkkoomme. Tähän tarkoitukseen asensimme Very Secure FTP Daemonin.

sudo apt-get install vsftpd /16/

FTP-ohjelman asentamisen lisäksi piti joitakin asetuksia muuttaa, jotta ohjelma toimisi tarkoituksen mukaisesti.

/etc/vsftpd.conf /16/

Ylläolevasta tiedostopolusta löytyvästä konfiguraatio tiedostosta piti seuraavat rivit ottaa pois kommenteista (#-merkki rivin edessä) tai muuttaa vastaavanlaiseksi. /16/

local_enable=YES
write_enable=YES
anonymous_enable=NO
chroot_local_user=YES

Testaamisen helpottamiseksi ajettiin vielä seuraava komento vsftp-daemoni käynnistyy uudestaan heti ja käynnistyksen yhteydessä ilman erillistä käynnistystä.

sudo /etc/init.d/vsftpd restart /16/

Palvelun luomiseksi tuli tehdä .service tiedosto, jossa määritellään palvelun nimi ja tyyppi ja muita tarpeellisia tietoja.

sudo gedit /etc/avahi/services/ftp.service /16/

Tähän tiedostoon tuli kirjoittaa seuraava koodi /16/ :

<?xml version="1.0" standalone='no'?>
<!DOCTYPE service-group SYSTEM "avahi-service.dtd">
<service-group>
<name>FTP file sharing</name> **Palvelun nimi.**
<service> **Tästä alkaa palvelun määrittely.**
<type>_ftp._tcp</type> **Palvelun tyyppi. Tässä tapauksessa FTP**
<port>21</port> **Portiksi määritellään 21, joka on FTP:lle määrätty portti.**
</service> **Tähän loppuu määrittely.**
</service-group>

4.4.3 Musiikkipalvelu

Musiikkipalvelun toteutimme Firefly Media Server:llä. Se on avoimen lähdekoodin ohjelma Roku SoundBridgelle tai iTunesille. Käytössä oli näiden kahden protokollat Roku Server Protocol tai Digital Audio Access Protocol. Käytössämme oli jäkimmäinen protokolla. /17/

Ohjelman asennuksen jälkeen jako oli automaattisesti päällä. Pystyimme hallitsemaan sitä selaimen kautta.

Firefly-ohjelmalla saimme musiikin jakoon DAAP-protokollalla. Yhteyden tähän jakoon saimme rhythmbox-ohjelmalla. Rhythmbox osoittautui käteväksi tähän tarkoitukseen, sillä se muisti jaot joihin se oli aikaisemmin ottanut yhteyden.

4.5 Soveltuvuudet käyttöön wlan mesh-verkoissa

Edellä esitetyistä protokollista parhaiten käyttöön wlan mesh-verkoissa sopisi työssämmekin käytetty applen protokollan implementaatio avahi. Miksi?

Avahi avoimen lähdekoodin sovelluksena on vapaasti kehiteltävissä omaan tarkoitukseen sopivaksi. Tässä työssä esitelty *SSDP* perustuu vanhentuneeseen *ITEF:än* internet luonnokseen, joten sen kehitystyö olisi vaivalloista ja tulisi maksamaan huomattavasti enemmän. *SLP* puolestaan odottaa standardin hyväksymistä.

Avahista on työtä kirjoittaessa vain linux-versio mutta sen kääntäminen käytettäväksi esimerkiksi windows-käyttöjärjestelmässä onnistunee. Alkuperäisestä applen bonjour-sovelluksesta on myös julkaistu windows versio apache-lisenssin alla. Omien tarpeidensa mukaan voi valita mieleisensä, sillä pohjimmiltaan molemmat ovat sama sovellus.

5. Mesh control panel

Projektin puitteissa teimme myös ohjelman, jolla käyttäjän olisi mahdollisimman helppo ottaa käyttöön wlan mesh-verkko. Ohjelma helpotti myös työtämme, sillä toistuvat konsoli komennot oli helppo ja nopeaa ajaa graafisen ohjelman valikoista. Ohjelma on koodattu python ohjelmointikielellä ja graafisen käyttöliittymään käytettiin pythonin tkinter-kirjastoa.

Ohjelman teko alkoi tietysti määrittelemällä, mitä kirjastojen osia ohjelmassa tarvitaan.

```
from Tkinter import * ← tkinter:stä otetaan käyttöön kaikki osat.
from ScrolledText import ScrolledText ← ScrolledText-kirjastosta otetaan
käyttöön vain sen ScrolledText-osa.
import tkSimpleDialog, tkMessageBox, tkFileDialog
import os, time
```

Ohjelmaikkunan teko aloitettiin luomalla tyhjä pohja nimellä root.

```
# Creates root window
root = Tk()
```

Seuraavaksi määriteltiin ohjelmaikkunan perusominaisuudet.

```
menu1 = Menu(root) ← Luodaan valikkopohja edellä luotuun pohjaan.
root.title("Mesh control panel") ← Otsikko palkkiin tuleva nimi.
root.geometry("250x250") ← Ohjelmaikkunan koko.
root.config(menu=menu1)
```

Valikoiden alle tulevan osan määrittely.

frame=Frame(root) ← ***Osoitetaan tämän osan kuuluvan root:iin.***
frame.pack(side=TOP) ← ***Tämä osa sijoitetaan päälimmäiseksi.***
ruutu = Label(frame, padx = 10, pady = 10, text="\n \nMesh Control Panel v0.3.6") ← ***Määriteltyyn osaan sijoitetaan ruutu nimellä teksti.***
ruutu.pack()

Valikko-rakenteen tekeminen oli itseään toistavaa, joten seuraavassa kuvataan vain yhden valikon rakenne.

connectmenu = Menu(menu1) ← ***Määritellään 'connectmenu':n kuuluvan edellä luotuun valikkopohjaan.***
menu1.add_cascade(label="Connect", menu=connectmenu) ← ***Connectmenu on alaspäin aukeavaa tyyppiä(cascade) ja sen otsikko on 'Connect'.***
connectmenu.add_command(label="Start mesh", command=start_mesh) ← ***Lisätään Connect-valikkoon 'Start mesh' valinta.***
connectmenu.add_separator() ← ***Väliviiva.***
connectmenu.add_command(label="Stop mesh", command=stop_mesh)
connectmenu.add_command(label="Quit", command=quit)

Valikon komennot olivat määritelty seuraavalla tavalla. Esimerkkinä toimii wlan mesh-verkon käynnistävä komento. Tässä voi huomioida hallintaohjelman edut. Komentoja on useita ja veisi paljon aikaa, joka kerta syöttää nämä komennot uudestaan, kun verkko pitäisi käynnistää. Lisäksi käyttö helpottuu, kun ei tarvitse tuntea kaikkia komentoja, vain muutama verkon määrittelevä tieto.

```
def start_mesh():
    os.system("killall NetworkManager") ← Lopetetaan ubuntun verkkoyhteyksin hallintaohjelma.
    os.system("modprobe zd1211rw") ← Otetaan wlan-sovittimen tarvitsema ajuri käyttöön.
    iface = tkSimpleDialog.askstring(" ", "Type your wlan interface:”) ← Kysytään wlan-sovittimen interface, jotta mesh-verkko voidaan liittää oikeaan laitteeseen.
    id = tkSimpleDialog.askstring(" ", "Type mesh id:") ← Kysytään mesh-verkolle tuleva nimi.
    redirectedGuiShellCmd("iw dev "+iface+" interface add mesh type mp mesh_id
    " + id) ← Luodaan mesh-verkko annettujen tietojen mukaan.
    os.system("ifconfig mesh up") ← Käynnistetään edellä luotu verkko.
    os.system("iwconfig mesh rate 54M auto") ← Asetaan mesh-verkolle wlanin maksimi nopeus, kuitenkin asettaen päälle automaattinen nopeuden säätö.
    os.system("avahi-autoipd mesh --daemon")  ← Otetaan käyttöön avahin tarjoama automaattinen ip-osoite ja laitetaan se pyörimään taustalla.
    time.sleep(5)
    tkMessageBox.showinfo("Start mesh", "Mesh network up!")
```

Ohjelmassa käytettiin lisäksi pop-up ikkunoita joidenkin komentojen tulosteiden näyttämiseen. Tähän tarkoitukseen käytettiin valmista koodia.

Ensimmäisessä koodissa määritellään pop-up ikkunan toiminta /18/ :

```
class GuiOutput:
    font = ('courier', 9, 'normal')
    def __init__(self, parent=None):
        self.text = None
        if parent: self.popupnow(parent)
    def popupnow(self, parent=None):
        if self.text: return
        self.text = ScrolledText(parent or Toplevel( ))
        self.text.config(font=self.font)
        self.text.pack( )
def write(self, text):
        self.popupnow( )
        self.text.insert(END, str(text))
        self.text.see(END)
        self.text.update( )
    def writelines(self, lines):
        for line in lines: self.write(line)
```

Tämä koodi ohjaa valikoiden koodin sisältämät konsoli komennot pop-up ikkunoihin / 18/ :

```
def redirectedGuiShellCmd(command):
    import os
    input  = os.popen(command, 'r')
    output = GuiOutput( )
    def reader(input, output):
        while True:
          line = input.readline( )
          if not line: break
          output.write(line)
    reader(input, output)
```

6. Tulevaisuudennäkymät

Tälläisen verkkoratkaisun tulevaisuus riippuu ihan siitä miten laitakanta kehittyy. Sillä työtä tehdessämme ainoa vaihtoehto oli ulkoinen wlan-sovitin ja laitekanta hyvin pieni. Laitteiden sisäiset sovittimet ovat yleistyneet siinä määrin, että niitä löytyy lähes jokaisesta pöytätietokoneestakin. Eikä näin ollen kovinkaan montaa kuluttajaa kiinnostaisi hankkia lisälaitetta, kun heidän tietokoneidensa sisällä on jo laite vastaavaan tarkoitukseen.

Wlan mesh-verkko luo säästö mahdollisuuksia, sillä sen muodostamassa verkossa ei tarvita lainkaan verkon perinteisiä laitteita. Säästöillä on kuitenkin rajansa, sillä uusien rakennusten suunnittelussa pitää ottaa huomioon wlan-verkkojen vaatimukset. Vanhoissa rakennuksissa saattaa tulla ongelmia verkon kattavuuden kanssa. Tietysti tässä kohdassa joku ajattelee, että lisätään ongelma kohtiin tukiasemia. Onhan se tietysti ratkaisu, mutta se ei ole enää mesh-periaatteen mukainen.

Tietoturva muodostaa oman ongelmansa, sillä perinteisten verkkolaitteiden puuttuessa liikenne kulkee jokaisen tietokoneen läpi tehden liikenteen kuuntelun helpommaksi. Työtämme tehdessä ei wlan mesh-verkkoon ollut tehty minkäänlaisia varsinaisia tietoturva ominisuuksia. Eräänlaisena tietoturva ominaisuutena toimi open 802.11s:n epävirallinen asema, sillä muodostettua mesh-verkkoa ei löytynyt ubuntun networkmanagerilla.

Kodeissa mahdollisia käyttökohteita olisivat erilaiset media-laitteet, jotka saattaisivat yleistyä helppokäyttöisyyden ansiosta. Laitteet ei tarvitsisi kuin kytkeä päälle ja ne olisivat yhteydessä toisiinsa. Hyllyjen takana vaanivista johto

Lähteet

Sähköiset lähteet

1 Wikipedia [www-sivu] Mesh networking [viitattu 4.5.2009] Saatavissa: http://en.wikipedia.org/wiki/Mesh_networking

2 Wikipedia [www-sivu] 802.11s [viitattu 4.5.2009] Saatavissa: http://en.wikipedia.org/wiki/802.11s

3 Open80211s [www-sivu] open80211s HOWTO [viitattu 27.7.2009] Saatavissa: http://o11s.org/trac/wiki/HOWTO

4 Open80211s [www-sivu] Building open80211s on ubuntu [viitattu 28.7.2009] Saatavissa: http://o11s.org/trac/wiki/Ubuntu

5 Jean Tourrilhes [www-sivu] iwconfig [viitattu 28.7.2009] Saatavissa: http://linux.die.net/man/8/iwconfig

6 Avahi-autoipd [man-sivu] Avahi-autoipd [viitattu 28.7.2009] Saatavissa: avahi-autoipd:n man-sivut.

7 Wikipedia [www-sivu] Service Discovery [viitattu 1.6.2009] Saatavissa: http://en.wikipedia.org/wiki/Service_discovery

8 Wikipedia [www-sivu] Zerconf [viitattu 28.7.2009] Saatavissa: http://en.wikipedia.org/wiki/Zeroconf

9 Stuart Cheshire, Marc Krochmal [tekstitiedosto] Multicast DNS [viitattu 1.6.2009]
Saatavissa: http://files.multicastdns.org/draft-cheshire-dnsext-multicastdns.txt

10 Wikipedia [www-sivu] SRV record [viitattu 4.6.2009] Saatavissa: http://en.wikipedia.org/wiki/SRV_record

11 The Avahi Team [www-sivu] Multimedia Applications [viitattu 28.7.2009] Saatavissa: http://avahi.org/wiki/MultimediaAvahiApplication

12 Stuart Cheshire, Marc Krochmal [tekstitiedosto] DNS-Based Service Discovery [viitattu 4.6.2009] Saatavissa: http://files.dns-sd.org/draft-cheshire-dnsext-dns-sd.txt

13 Wikipedia [www-sivu] Simple Service Discovery Protocol [viitattu 8.6.2009] Saatavissa: http://en.wikipedia.org/wiki/Simple_Service_Discovery_Protocol

14 Yaron Y. Goland, Ting Cai, Paul Leach, Ye Gu [tekstitiedosto] Simple Service Discovery Protocol/1.0 [viitattu 8.6.2009] Saatavissa: http://quimby.gnus.org/internet-drafts/draft-cai-ssdp-v1-03.txt

15 Erik Guttman, Charles Perkins, John Veizades, Michael Day [tekstitiedosto] Service Location Protocol, Version 2 [viitattu 10.6.2009] Saatavissa: http://tools.ietf.org/id/draft-ietf-svrloc-protocol-v2-15.txt

16 Ubuntuforums [www-sivu] HOWTO Ridiculously easy home file sharing with FTP and Zeroconf [viitattu 28.7.2009] Saatavissa: http://ubuntuforums.org/showthread.php?t=218630

17 Wikipedia [www-sivu] Firefly Media Server [viitattu 28.7.2009] Saatavissa: http://en.wikipedia.org/wiki/Mt-daapd

18 CodeIdol Labs [www-sivu] GuiStreams: Redirecting Streams to Widgets [viitattu 23.6.2009] Saatavissa: http://codeidol.com/python/python3/GUI-Coding-Techniques/GuiStreams-Redirecting-Streams-to-Widgets/

Liitteet

Liite 1: Mesh Control Panel koodi kokonaisuudessaan.