Asko Paavilainen

Valaisinjärjestelmän hallitseminen Ethernetin avulla

Sähkötekniikan korkeakoulu

Diplomityö, joka on jätetty opinnäytteenä tarkastettavaksi diplomi-insinöörin tutkintoa varten Espoossa 15.5.2016.

Työn valvoja:

Prof. Seppo Ovaska

Työn ohjaaja:

DI Kai Kauto

AALTO-YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Asko Paavilainen

Työn nimi: Valaisinjärjestelmän hallitseminen Ethernetin avulla

Päivämäärä: 15.5.2016 Kieli: Suomi Sivumäärä: 8+51

Sähkötekniikan ja automaation laitos

Professuuri: Teollisuuselektroniikka

Työn valvoja: Prof. Seppo Ovaska

Työn ohjaaja: DI Kai Kauto

Tämän diplomityön tarkoitus on tutkia, miten valaisinjärjestelmää voidaan hallita Ethernetin avulla. Työ jakautuu kahteen pääosioon. Ensimmäinen osio keskittyy Ethernetin, TCP/IP:n ja teollisten protokollien teoreettiseen taustaan OSI-viitekehyksen valossa. Toinen osa keskittyy tarvittaviin kehitystyökaluihin ja ohjelmistoihin. Tätä varten työssä kehitettiin kaupalliseen kehitysalustaan ja avoimen lähdekoodin lwIP-pinoon perustuva esimerkkijärjestelmä. Atollic TrueSTUDIO valittiin sopivimmaksi kehitysympäristöksi. LwIP-pino todettiin toiminnallisuuden puolesta sopivaksi, mutta dokumentaationsa ja tukensa vuoksi sen käyttöä myöhemmissä tuotteissa ei suositella. Vaikka työssä kehitetty laitteisto täyttää sille asetetut tavoitteet, työssä havaitaan osoitteisiin liittyvä todellisessa käyttöympäristössä ilmenevä ongelmia, joita ei saatu ratkaistuksi pelkän TCP/IP-protokollaperheen avulla. Ethernetin kapasiteetin havaittiin ylittävän valaisinjärjestelmän vaatimukset.

Avainsanat: Teollinen Ethernet, valaistus, TCP/IP, lwIP

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Asko Paavilainen

Title: Controlling a Luminaire System via Ethernet

Date: 15.5.2016 Language: Finnish Number of pages: 8+51

Department of Electrical Engineering and Automation

Professorship: Industrial Electronics

Supervisor: Prof. Seppo Ovaska

Advisor: M.Sc. (Tech.) Kai Kauto

The purpose of this thesis is to investigate what it takes to control a luminaire system via Ethernet. The thesis focuses in two major topics. The first topic examines the theory of Ethernet, TCP/IP and industrial protocols and compares them to the OSI reference model. The second topic focuses on required developing tools and available software. An example system was created using a commercially available development board and an open source TCP/IP stack called lwIP. The results suggest that Atollic TrueSTUDIO should be used as the developing environment. The lwIP stack is revealed to be functionally acceptable. However, due to its documentation and support, it is not recommended for future products. Some potential issues that cannot be solved using bare TCP/IP are identified. The capacity of Ethernet is found to exceed the requirements of a luminaire system.

Keywords: Industrial Ethernet, lighting, TCP/IP, lwIP

Esipuhe

Tahdon kiittää valvojaani prof. Ovaskaa, työtä ohjannutta DI Kautoa, sekä häntä parhaansa mukaan avustanutta esimiestäni Matti Alavaa työn aikana saamastani tuesta. Lisäksi kiitos kuuluu kollegoilleni, joista Juhalle erityinen kiitos lähdemateriaalin lainasta, sekä laskuseurueille tarpeeseen tulleista irtiotoista.

Myös menneistä vuosista on kiitettävä joitakin henkilöitä. Ilkalle kiitos niin yhteisistä projekteista, kurssien lempinimistä ja lokajoulusta, kuin opintojen ulkopuolisestakin ajasta. Petterille kiitos yhteisistä todennäköisyyslaskennan luennoista ja kotitehtävistä, joita molempia oli lopulta yllättävänkin paljon. Onnistuit todistamaan kaksi asiaa: todennäköisyyksillä ei ole mitään tekemistä todellisuuden kanssa, eikä ikuisia pelikortteja ole olemassa. Eikä sovi myöskään unohtaa vanhempiani, te mahdollistitte omalta osaltanne tämän kaiken. Kiitos siitä.

Lahdessa, 15.5.2016

Asko Paavilainen

Sisällysluettelo

Lyhenteet

APIPA	Menetelmä, jolla verkon IP-osoitteista voidaan sopia ilman palvelinta (*Automatic Private IP Addressing*)
ARP	Protokolla, jolla selvitetään IP-osoitetta vastaava MAC-osoite (*Address Resolution Protocol*)
AUI	10 Mb/s verkoissa käytetty MAC- ja PHY-piirien rajapinta (*Attachment Unit Interface*)
CAN	Eräs kenttäväylätyyppi (*Controller Area Network*)
CC	EtherCAT:n soveltuvuusluokka (*Conformance Class*)
CIP	Yleinen teollinen protokolla (*Common Industrial Protocol*)
CRC	Tarkisteiden laskemiseen tarkoitettu algoritmi (*Cyclic Redundancy Check*)
CSMA/CD	IEEE:n mukainen nimitys Ethernet-verkolle (*Carrier Sense Multiple Access / Collision Detection*)
DNS	Järjestelmä, joka muuntaa verkko-osoitteita IP-osoitteiksi (*Domain Name System*)
DHCP	IP-osoitteiden dynaamiseen määrittämiseen käytettävä protokolla (*Dynamic Host Configuration Protocol*)
DIX	Yksi varhaisista Ethernetin versioista, nimetty kehittäjiensä (DEC, Intel ja Xerox) mukaan
DSAP	Kohteen palvelupiste (*Destination Service Access Point*)
EDS	EtherNet/IP:n mukainen sähköinen datalehti (*Electronic Data Sheet*)
ENI	EtherCAT-verkkoa kuvaava tiedosto (*EtherCAT Network Information*)
ESI	EtherCAT-laitetta kuvaava tiedosto (*EtherCAT Slave Information*)
GMII	1000 Mb/s verkoissa käytetyn MAC- ja PHY piirien välinen rajapinta (*Gigabit Media Independent Interface*)
IANA	Järjestö, joka vastaa mm. IP-osoitteiden maailmanlaajuisesta jakautumisesta (*Internet Assigned Numbers Authority*)
ICMP	TCP/IP-verkon sisäiseen toimintaan liittyvä protokolla (*Internet Control Message Protocol*)
IEEE	Mm. standardeja määrittelevä kansainvälinen tekniikan alan järjestö (*Institute of Electrical and Electronics Engineers*)
IGMP	Monilähetyksiin käytettävä protokolla (*Internet Group Management Protocol*)
IP	TCP/IP-perheen alimman tason protokolla (*Internet Protocol*)
IPv4	IP-protokollan versio 4 (*Internet Protocol Version 4)*
IPv6	IP-protokollan versio 6 (*Internet Protocol Version 6)*
ISO	Kansainvälinen standardoimisjärjestö (*International Organization for Standardization*)
LLC	Korkein IEEE 802.3:n määrittelemä taso (*Logical Link Control*)
lwIP	Eräs TCP/IP-pino (*light weight TCP/IP stack*)
MAC	Siirtotien haltuunotto, tai yksi TCP/IP:n mukaisista kerroksista (*Media Access Control*)
MA-L	Suuri MAC-osoitteiden lohko (*MAC Address Block Large*)

MA-M	Keskikokoinen MAC-osoitteiden lohko (*MAC Address Block Medium*)
MA-S	Pieni MAC-osoitteiden lohko (*MAC Address Block Small*)
MII	MAC- ja PHY-kerrosten välinen rajapinta(*Media Independent Interface*)
ODVA	DeviceNet:iä hallinnoiva yhtiö (*Open DeviceNet Vendor Association*)
OUI	Yksilöllinen organisaation tunniste (*Organizationally Unique Identifier*)
OSI	Yleinen tiedonsiirron viitekehys (*Open Systems Interconnection*)
PI	Järjestö, joka määrittelee ja ylläpitää PROFIBUS- ja PROFINET-standardeja (*PROFIBUS & PROFINET International*)
PCB	LwIP-pinon käyttämä nimitys protokollan hallintaan liittyvästä rakenteesta (*Protocol Control Block*)
PWM	Pulssinleveysmodulaatio (*Pulse Width Modulation*)
PHY	Ethernetin fyysiseen kerrokseen kuuluva mikropiiri
PID	Protokollan tunniste (*Protocol Identifier*)
RMII	Yksi MAC- ja PHY-kerrosten rajapinnoista (*Reduced Media Independent Interface*)
SNAP	LLC-kehyksen erikoistapaus (*Subnetwork Access Protocol*)
SSAP	Lähteen palvelupiste (*Source Service Access Point*)
STP	Suojattu kierretty pari (*Shielded Twisted Pair*)
TCP	Toinen TCP/IP-protokollaperheen kuljetuskerroksen protokollista (*Transmission Control Protocol*)
TCP/IP	Protokollaperhe, joka on nimetty TCP- ja IP-protokollien mukaan
UDP	Toinen TCP/IP-protokollaperheen kuljetuskerroksen protokollista (*User Datagram Protocol*)
UTP	Suojaamaton kierretty pari (*Unshielded Twisted Pair*)

1 Johdanto

Tämä diplomityö on tehty Teknoware Oy:lle, joka on kotimainen julkisen liikenteen valaistusjärjestelmiin sekä kiinteistöjen ja laivojen turvavalaistusjärjestelmiin erikoistunut yritys. Työ käsittelee raideliikenteen sisävalaistuksen hallitsemista Ethernet-väylän avulla. Tyypillinen valaisinjärjestelmä muodostuu vaunukohtaisesta ohjausyksiköstä ja enintään muutamasta kymmenestä valaisimesta. Tässä työssä keskitytään järjestelmän hallitsemiseen kokonaisuutena, ja ohjausyksikön ja valaisinten välinen rajapinta on rajattu työn ulkopuolelle.

Valaistustekniikka on viime vuosina kehittynyt valtavasti. Hehkulamppujen tilalle on kehitetty energiatehokkaampia vaihtoehtoja, kuten loistelamppuja ja LED-valaisimia. Energiatehokkuusvaatimusten myötä energiansäästölamppujen kehitys ja markkinat ovat kasvaneet entisestään, sillä esimerkiksi Yhdysvalloissa ja EU:n alueella energiatehokkuusvaatimukset rajoittavat hehkulamppujen myyntiä. Aluksi hehkulamppuja korvattiin pienloistelampuilla, joiden valotehokkuus on hehkulamppuihin verrattuna karkeasti nelinkertainen. Pienloistelamput eivät kuitenkaan ole aivan ongelmaton ratkaisu, koska ne sisältävät mm. elohopeaa, ja niiden kirkkauden säätäminen on hankalaa. Nykyään LED-teknologia valtaa markkinoita nopeasti. LEDien himmentäminen on helpompaa kuin pienloistelamppujen, eivätkä ne sisällä elohopeaa. Lisäksi LEDien kehitystyö jatkuu edelleen ja niiden valotehokkuuden odotetaan nousevan pienloistelamppuihin verrattuna moninkertaiseksi. LEDien eliniän vertaaminen muihin valaisintyyppeihin on hieman hankalaa, koska LEDit eivät tyypillisesti rikkoudu hehkulamppujen tavoin, vaan niiden tuottama valon määrä vähenee hiljalleen. Yleisesti LED-valon eliniän katsotaan päättyneen, kun sen valotehosta on jäljellä 70 %. Tyypillisesti LED-valmistajat lupaavat tämän tapahtuvan n. 50 000 tunnin käytön jälkeen, mikä vastaa n. 5,7 vuoden yhtäjaksoista käyttöä. [1]

Energiansäästötrendin viedessä valaistusalaa eteenpäin myös ohjausjärjestelmät ovat kehittyneet valtavasti. Vaikka valaisimien valotehokkuus on kasvanut uusien valonlähteiden myötä huomattavan paljon, energiaa voidaan säästää myös säätämällä valonlähteiden kirkkautta ja sitä kautta niiden kuluttamaa tehoa. Järjestelmät jakautuvat kahteen päätyyppiin: manuaalisiin ja automaattisiin järjestelmiin. Myös erityyppisten järjestelmien yhdistäminen on mahdollista, jolloin järjestelmä sisältää molemmille tyypillisiä ominaisuuksia. Manuaalisissa järjestelmissä valaisimien säätäminen on käyttäjän vastuulla. Automaattisiin järjestelmiin taas kytketään erilaisia antureita, joiden perusteella järjestelmä säätää valaistusta. Läsnäoloanturilla voidaan havaita onko tila tyhjä ja säästää energiaa sammuttamalla valaistus kokonaan tai himmentämällä sitä merkittävästi silloin, kun tilassa ei ole ketään. Valoanturien avulla taas voidaan hyödyntää auringon tuottamaa valoa ja himmentää valaisimia silloin, kun päivänvaloa on riittävästi. Energiansäästö ei ole päivänvaloa hyödyntävien järjestelmien ainoa etu, sillä auringon tuottamalla valolla on myös psykologinen vaikutus, ja se koetaankin keinotekoista valoa miellyttävämmäksi. [1, 2]

Valaisimien elinkaarikustannukset koostuvat kulutetun energian lisäksi investointi- ja huoltokustannuksista. Asiakkailta saadun tiedon mukaan valaistuksen energiansäästö

on raideliikenteessä toissijaista, mutta huoltokustannukset sitäkin olennaisempi tekijä. Tämä on ymmärrettävää, sillä kulkuneuvon ollessa huollettavana, se ei tee tuottoa omistajilleen, ja sekoittaa aikatauluja. Eliniän maksimointi ja valaisimien valvonta ovatkin järjestelmän tärkeimpiä tehtäviä. Turvallisuussyistä valaisimia ei milloinkaan sammuteta, mutta pelkkä himmentäminenkin laskee LEDien sisäistä lämpötilaa ja sitä kautta pidentää sen elinikää. Julkisten kulkuneuvojen kilpaillessa asiakkaistaan myös matkustusmukavuus on tärkeää. Koska julkisessa kulkuneuvossa on useita matkustajia, säätämisen on tapahduttava keskitetysti, esimerkiksi kuljettajan tai konduktöörin toimesta. [2]

Teknowaren valaisinjärjestelmissä on ennenkin hyödynnetty mm. CAN-väylää. Järjestelmän ohjaamiseen tarvittava datamäärä on nykyaikaisten tietoverkkojen kapasiteetti huomioiden melko vähäistä, ja siksi tarve Ethernetin käytölle ei nousekaan valaisinjärjestelmän puolelta. Asiakkaiden tiedusteluista on kuitenkin havaittu, että järjestelmän hallitsemiselle Ethernet-väylän kautta olisi kiinnostusta. Ethernetin etuna on se, että sitä käytetään esimerkiksi junissa muihinkin tarkoituksiin, jolloin verkko on jo valmiiksi olemassa. Valmiin, junassa käytettävän järjestelmän kehittäminen ei tämän työn puitteissa ole kuitenkaan mahdollista, koska käytettävät protokollat ovat usein valmistajakohtaisia, eikä niiden dokumentaatiota ole julkisesti saatavilla. Työn tavoitteena onkin perehtyä Ethernet-väylän toimintaan ja sen päällä yleisimmin käytettäviin protokolliin. Teollisista protokollista tarkasteluun on valittu Modbus/TCP, EtherNet/IP, EtherCAT sekä PROFINET. Toisena tavoitteena on kerätä kokemusta lwIP-pinosta ja käytettävistä kehitystyökaluista, sekä arvioida niiden soveltuvuutta myöhemmin kehitettäviin tuotteisiin.

1.1 Työn rakenne

Työ rakentuu muutamasta erillisestä kokonaisuudesta. Sen ensimmäisessä osassa, luvussa 2, käsitellään tiedonsiirtoa yleisellä tasolla. Luvun keskeisin aihe on OSI-viitekehys, jota käytetään useiden eri tiedonsiirtomenetelmien mallina. Luku 3 keskittyy Ethernetiin, jonka toimintaperiaate käydään läpi OSI-viitekehyksen valossa. Käsittely aloitetaan siirtokerrokselta ja sen määrittämistä kehyksistä, minkä jälkeen käsitellään kehysten muuttamista fyysisen kerroksen signaaleiksi, sekä Ethernetissä käytettäviä piiritason ratkaisuja. Tarkempi käsittely on rajattu Ethernetin parikaapelipohjaisiin versioihin. Luku 4 käsittelee Internetin selkärangan muodostavaa TCP/IP-protokollaperhettä, minkä jälkeen teollisia Ethernet-protokollia käsitellään luvussa 5.

Työn soveltavassa osuudessa, luvussa 6, suunniteltiin valaisinten esittelykäyttöön soveltuva ohjauslaite. Koska valaisinten ja ohjainlaitteen välinen rajapinta rajattiin työn ulkopuolelle, työssä käytettiin yksinkertaistettua järjestelmää, joka koostuu ohjainlaitteen lisäksi pulssinleveysmodulaatiolla ohjattavasta viisikanavaisesta LED-valonlähteestä. Järjestelmän ohjaamiseen kehitettiin tietokonesovellus ja graafinen käyttöliittymä. Sen tekninen toteutus rajattiin työn ulkopuolelle, ja ohjelmaa käsitellään työssä ainoastaan käyttöliittymän ja sen tuottaman verkkoliikenteen kautta.

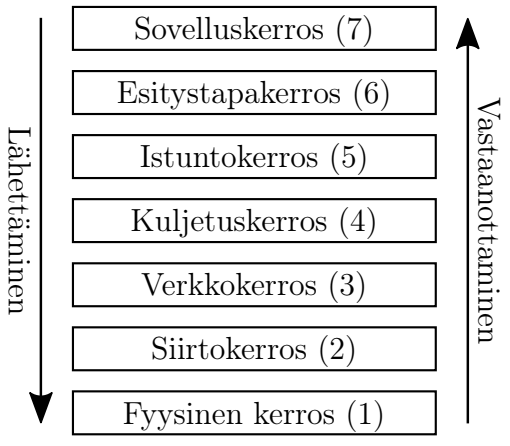
2 Tiedonsiirron periaatteita

Jotta Ethernetin käsitteleminen olisi yksinkertaisempaa, on syytä käsitellä ensin joitakin tiedonsiirron periaatteita yleisellä tasolla. Tämän luvun tarkoitus on erottaa yleisellä tasolla käsiteltävissä olevat aiheet seuraavissa luvuissa käsiteltävistä yksityiskohtaisemmista aiheista. Luvun keskeisin aihe on tiedonsiirtomenetelmien referenssinä käytettävä OSI-viitekehys.

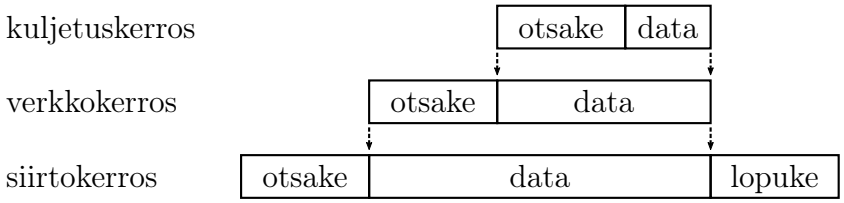
2.1 OSI-viitekehys

OSI on kansainvälisen standardoimisjärjestön, ISO:n, vuonna 1978 luoma tiedonsiirtoyhteyden arkkitehtuurimalli. Se jakaa yhteyden seitsemään toisistaan riippumattomaan tasoon, jotka ovat kuvan 1 mukaisesti sovelluskerros, esitystapakerros, istuntokerros, kuljetuskerros, verkkokerros, siirtokerros ja fyysinen kerros. Tasot numeroidaan numeroilla 1-7 siten, että pienempi numero tarkoittaa alempaa tasoa. Jokaiselle tasolle on määritelty tietty tarkoitus sekä rajapinnat ylempään ja alempaan tasoon. Toisistaan riippumattomien kerrosten myötä jokainen kerros kommunikoi näennäisesti suoraan toisen samanlaisen kerroksen kanssa. Tasot 1-3 liittyvät verkon toimintaan, kun taas tasot 5-7 liittyvät sovelluksen toimintaan. [3, 4]

Fyysinen kerros on OSI-viitekehyksen alin taso. Se vastaa datan siirtämisestä fyysisten sijaintien välillä määrittelemällä yhteyden fyysisen rajapinnan. Lähetettäessä fyysinen kerros muuntaa ylemmältä tasolta saamansa datan sähköisiksi signaaleiksi, ja vastaavasti vastaanotettaessa tulkitsee sähköiset signaalit digitaaliseksi dataksi. Jotta väylän laitteet olisivat sähköisesti ja mekaanisesti yhteensopivia, fyysisen kerroksen on määriteltävä mm. signaalien taso, linjakoodaus, käytettävät liittimet ja kaapelit, sekä signaalien synkronointi laitteiden välillä. Huomattavaa on, että esimerkiksi kaapelit eivät ole osa fyysistä kerrosta, se ainoastaan määrittelee ne. [3, 5]



Kuva 1: OSI-viitekehys koostuu seitsemästä toisistaan riippumattomasta tasosta. Dataa lähetettäessä siirrytään aina ylemmältä tasolta alemmalle, ja vastaanotettaessa alemmalta ylemmälle.



Kuva 2: Ylemmän tason kehys kapseloidaan alempien tasojen kehyksiin. Näin tasot pysyvät toisistaan riippumattomina.

Siirtokerros luo, lähettää ja vastaanottaa paketteja, joita kutsutaan kehyksiksi. Siirtokerroksen kehys kapseloi verkkokerrokselta saamansa tiedot omaan kehykseensä kuvan 2 mukaisesti. Kehykseen kuuluu verkkokerrokselta saadun osuuden lisäksi otsake ja lopuke. Otsakkeesta selviää esimerkiksi vastaanottajan ja lähettäjän osoitteet. Lopuke sisältää yleensä virheiden tunnistamiseen käytettävän tarkistuskentän. [3, 5]

Verkkokerros huolehtii viestien reitittämisestä. Se muuntaa tarvittaessa verkko-osoitteet laitteisto-osoitteisiin ja pilkkoo suuret datamäärät useampiin kehyksiin. Verkkokerros lisää lähetettävään dataan siirtokerroksen tapaan oman otsakekenttänsä. Myös lopukkeen käyttäminen on mahdollista, mutta tyypillisesti verkkokerroksessa ei käytetä omaa lopuketta. Mikäli tiedonsiirtoyhteydessä käytetään reitittimiä, kehys puretaan niissä verkkokerrokselle asti. Reititin muodostaa verkkokerroksen kehyksen uudelleen, ja lähettää sen seuraavalle vastaanottajalle. [3, 5]

Kuljetuskerros huolehtii yhteyksistä ja niiden avaamisesta. Se varmistaa, että yhteys toimii hyväksyttävällä varmuudella ja nopeudella. Kuljetuskerroksen otsakkeeseen lisätään mm. kehyksen järjestysnumero, jonka avulla vastaanottava laite tietää, että kaikki viestit saapuvat perille oikeassa järjestyksessä. Lisäksi Kuljetuskerros voi huolehtia virheenkorjauksesta, mikäli alemmat kerrokset eivät tee sitä. Kuljetuskerros on tärkeä rajapinta sovellusten ja verkon välillä, sillä se tarjoaa verkon toiminnallisuuden sovelluspohjaisille kerroksille. [3, 5, 6]

Istuntokerroksen tehtävä on huolehtia sovellusten välisestä tiedonsiirrosta. Alemmat kerrokset tarjoavat sille luotettavan yhteyden verkkoon, kun taas istuntokerros on alin taso, jolla sovellukset kommunikoivat. Esimerkiksi jaetun resurssin avaaminen ja sulkeminen sekä tarvittaessa tunnistautuminen ovat istuntokerroksen vastuulla. Istuntokerroksen yläpuolella on esitystapakerros, joka määrittelee, kuinka lähetettävä tieto on koodattu. Tieto voidaan tarvittaessa myös salata. OSI-viitekehyksen ylin kerros on nimeltään sovelluskerros. Tällä ei tarkoiteta käyttäjälle näkyvää sovellusta, vaan kyseessä on suoritettavan sovelluksen ja verkon välinen rajapinta. Sovelluskerros tarjoaa käyttäjän sovellukselle korkean tason palveluita, joita voivat olla esimerkiksi sähköposti tai tiedoston siirtäminen. [5, 6, 7]

2.2 Kaksisuuntaisuus ja vuorosuuntaisuus

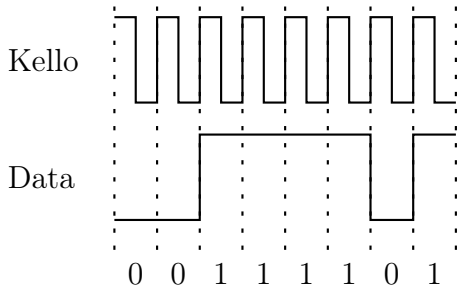
Mikäli minkä tahansa verkon laitteet kommunikoivat molempiin suuntiin, on tiedonsiirron oltava joko kaksisuuntaista (englanniksi *full-duplex*) tai vuorosuuntaista (englanniksi *half-duplex*). Vuorosuuntaisessa kommunikoinnissa osapuolet jakavat käyttämänsä siirtotien. Näin ollen osapuolet eivät voi lähettää dataa samanaikaisesti, tai niiden viesti törmäävät. Vuorosuuntaisessa tiedonsiirrossa tarvitaankin jokin menetelmä, jolla yksittäinen laite saa siirtotien varattua omaan käyttöönsä.

Kaksisuuntaisessa tiedonsiirrossa datan lähettämiseen ja vastaanottamiseen käytetään eri siirtoteitä, jolloin laitteet voivat lähettää ja vastaanottaa samanaikaisesti. Tällöin viestit eivät voi edes teoriassa törmätä toisiinsa, ja vuorojen jakaminen ei ole tarpeen. Kaksisuuntaisella tiedonsiirrolla saadaankin hyödynnettyä verkon kapasiteetti paremmalla hyötysuhteella. [4, 6]

2.3 Signaalien tahdistaminen

Jotta verkon solut voivat viestiä toisilleen, niiden on kyettävä erottamaan väylän signaalista yksittäisiä bittejä. Tätä varten niiden on tahdistettava kommunikoinnissa käytettävät signaalit. Tiedonsiirto voi olla joko synkronista tai asynkronista. Asynkroninen tiedonsiirto perustuu ennalta sovittuun siirtonopeuteen. Vastaanottaja tunnistaa ensimmäisen siirrettävän bitin ja alkaa näytteistää dataväylää merkittävästi tiedonsiirrossa käytettävää taajuutta korkeammalla taajuudella, tyypillisesti 16-kertaisella taajuudella. Näytteistyksen jälkeen bittien sijainti päätellään sovitun siirtonopeuden avulla.

Synkroninen tiedonsiirto taas perustuu kellosignaalin käyttöön. Se voi olla kokonaan erillinen signaali, jolloin siirrettävien signaalien määrä kasvaa. Toinen vaihtoehto on sisällyttää kellosignaali datasignaaliin. Synkronisen lähetyksen alussa käytetään aloituskuviota, joka antaa vastaanottajalle aikaa tahdistua lähetykseen. Tämän jälkeen tahdistuminen saadaan ylläpidettyä kellosignaalin avulla. Synkronisella tiedonsiirrolla voidaankin saavuttaa nopeampi ja tehokkaampi yhteys kuin asynkronisella tiedonsiirrolla. Kuvassa 3 on esitetty esimerkki synkronoidusta tiedonsiirrosta. [4, 7]



Kuva 3: Esimerkki synkronoidusta tiedonsiirrosta. Tässä esimerkissä käytetään erillistä kellosignaalia. Data näytteistetään kellosignaalin laskevalla reunalla ja asetetaan nousevalla reunalla.

2.4 Yhteydellinen ja yhteydetön tiedonsiirto

Yhteydetön protokolla lähettää ylemmältä tasolta saamansa viestin välittömästi eteenpäin. Järjestysnumeroita ei käytetä, eikä yhteyden virheettömyyttä voida taata. Viestit voivat kulkea eri reittejä pitkin ja saapua perille eri järjestyksessä kuin ne on lähetetty. Virheentunnistusta voidaan käyttää, mutta virheelliseksi havaittua dataa ei lähetetä uudelleen, vaan se ainoastaan hylätään. Yhteydettömän protokollan datapakettia kutsutaan sähkeeksi, jota vastaa englanninkielinen termi *datagram*.

Yhteydellisessä protokollassa yhteys on avattava ennen tiedonsiirron aloittamista. Käytännössä se tarkoittaa, että molemmat osapuolet käynnistävät yhteyttä varten tilakoneen, joka siirtyy tilasta toiseen esimerkiksi datan lähettämisen, vastaanottamisen tai odotusajan umpeutumisen seurauksena. Yhteydellinen protokolla voi tarjota virheettömän palvelun, mutta yhteyden avaaminen, sulkeminen ja ylläpitäminen vievät aikaa, minkä seurauksena yhteydellinen protokolla on yhteydetöntä protokollaa raskaampi. [4]

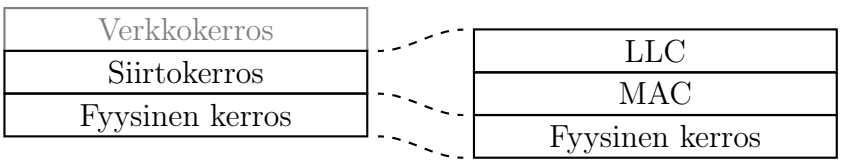
3 Ethernet

Nykyään termillä Ethernet viitataan yleensä tietokoneiden väliseen lähiverkkoon, joka määritellään IEEE:n standardissa 802.3. Termin historia yltää kuitenkin aina 1970-luvun puoliväliin asti. Se otettiin ensimmäisen kerran käyttöön Havaijin yliopistossa, jossa avattiin kampusten välinen ALOHA-verkko. ALOHA käytti siirtotienään "eetteriä"(englanninkieliseltä nimeltään *ether*) ja se yhdisti kampukset verkkoon (englanniksi *network*), joten sitä alettiin puhekielessä kutsua Ethernetiksi. Xerox Corporation jatkoi ALOHA:sta alkanutta kehitystyötä ja 1980-luvulla se julkaisi yhdessä Digital Equipment Corporationin ja Intelin kanssa ensimmäisen Ethernet-spesifikaation, joka tunnettiin nimellä *Ethernet Blue Book 1*. Se korvattiin myöhemmin uudella versiolla, joka tunnetaan nimillä *Ethernet Blue Book 2*, *DIX* ja *Ethernet II*. IEEE julkaisi vuonna 1983 Ethernet II:een perustuvan standardin 802.3. Teoreettisella tasolla IEEE 802.3 määrittelemä verkko on nimeltään CSMA/CD, mutta käytännössä sekä 802.3:n että Ethernet II:n mukaisia verkkoja kutsutaan yleisesti Ethernetiksi. Tässä työssä Ethernetillä tarkoitetaan nimenomaan IEEE:n standardoimaa verkkoa, ellei asiasta erikseen mainita.

Ethernet sijoittuu OSI-viitekehyksessä tasoille 1-2, kattaen fyysisen kerroksen ja siirtokerrokseen. Tätä ylemmät kerrokset ovat Ethernetin päällä toimivia protokollia, joita käsitellään luvuissa 4 ja 5. Lisäksi Ethernet jakaa sekä fyysisen kerroksen että siirtokerroksen useampaan tasoon. Tasojen määrä riippuu käytettävästä versiosta. Ethernetin historian tunteminen onkin tarpeen, sillä sen vaikutus näkyy Ethernetissä edelleen. [3, 6]

3.1 Siirtokerros

IEEE:n 802.3 jakaa siirtokerroksen kuvan 4 mukaisesti kahteen osaan. Ylempi kerros on nimeltään LLC ja alempi MAC. LLC:n tehtävä on luoda täysin verkon tyypistä riippumaton rajapinta kaikille 802-perheen verkoille. Sen myötä fyysisenä kerroksena voidaan käyttää esimerkiksi Ethernetiä tai langatonta WLAN-verkkoa. MAC-kerroksen tehtävä on määritellä siirrettävä kehys, verkossa käytettävät osoitteet sekä huolehtia kehyksen siirtämisestä ja törmäysten tunnistamisesta. [6]



Kuva 4: IEEE 802.3 jakaa siirtokerroksen kahteen osaan. [6]

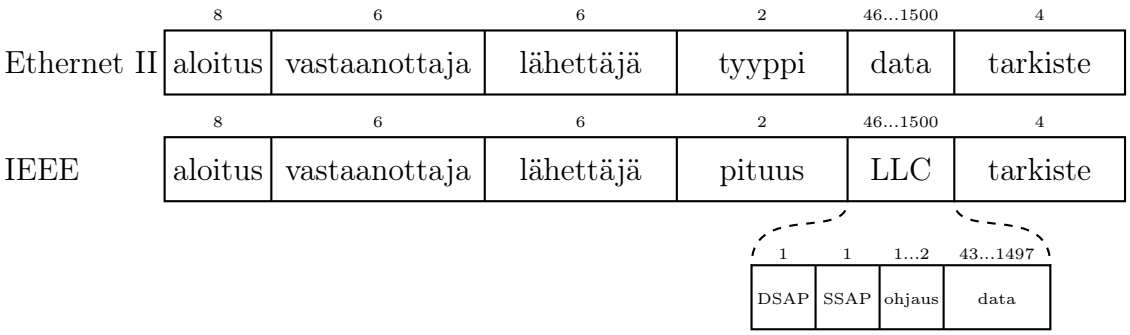
3.1.1 Ethernet-kehys

Ethernet II määrittelee kehyksen, joka koostuu kuvan 5 mukaisesti aloituskuviosta, vastaanottajan ja lähettäjän osoitteista, käytettävän protokollan määrittelevästä tyyppikentästä, datasta ja tarkisteesta. IEEE:n alun perin määrittelemä kehys poikkeaa kuitenkin Ethernet II -kehyksestä. Se korvaa tyyppikentän kehyksen pituudella, sekä datan LLC-tason rakenteella. Muilta osin kehykset ovat yhdenmukaisia. Myös IEEE:n määrittelemä kehys on esitetty kuvassa 5. Tyyppikentän mahdolliset arvot on määritetty siten, että eri standardien mukaiset kehykset voidaan erottaa toisistaan. Myöhemmin IEEE salli myös Ethernet II -tyyppisen kehyksen käyttämisen 802.3-standardin mukaisessa verkossa.

Aloituskenttä muodostuu kahdeksasta tavusta vuorottelevia ykkösiä ja nollia sillä poikkeuksella, että kaksi viimeistä bittiä ovat aina ykkösiä. Vuorottelevat bitit antavat vastaanottajalle aikaa tahdistamista varten, ja kaksi viimeistä bittiä merkitsevät aloituskentän päättymisen. Se ei sisällä hyödyllistä dataa, eikä sitä välitetä vastaanotettaessa siirtokerrokselle. Ensimmäiset tietoa sisältävät kentät kertovat vastaanottajan ja lähettäjän MAC-osoitteet. Mikäli vastaanottajana on yksittäinen laite, osoitekentän ensimmäinen bitti on nolla. Ryhmälähetyksen tapauksessa kyseinen bitti on ykkönen.

Osoitteiden jälkeen on vuorossa kehystyypistä riippuen joko tyyppikenttä tai pituuskenttä. Molemmat ovat kahden tavun mittaisia. Ethernet II -tyyppisten kehysten tyyppikentän numeerinen arvo on aina vähintään 1536 eli 0x0600. Tyyppikenttä kertoo, minkä protokollan mukaista dataa kehys kuljettaa. Esimerkiksi arvo 0x0800 tarkoittaa IPv4 protokollaa ja 0x86DD tarkoittaa IPv6 protokollaa. IEEE:n määrittelemän kehyksen pituus kertoo LLC-kentän koon. Sen maksimipituus on 1500 tavua, joten kehystyypit voidaan tunnistaa helposti toisistansa.

Kun käytettävä kehystyyppi on tunnistettu, päästään tiedonsiirron ytimeen – siirrettävään dataan. Datakentän minimipituus on 46 tavua ja maksimipituus 1500 tavua. Väärän pituiset kehykset hylätään. Siksi liian lyhyisiin kehyksiin lisätään ylimääräistä dataa minimipituuden täyttämiseksi. Ethernet II -kehys ei ota kantaa



Kuva 5: Ethernet II ja IEEE 802.3 määrittelevät hieman toisistaan poikkeavat kehykset. Kunkin kentän päälle on merkitty kyseisen kentän pituus tavuina. IEEE:n LLC-rakenne kapseloidaan MAC-tason rakenteeseen. [6]

Kuva 6: SNAP-kehys muodostuu normaalien LLC-kenttien lisäksi SNAP OIU-kentästä ja SNAP PID -kentästä. Kenttien päälle merkitty numero kertoo kentän koon tavuina. [6]

datakentän sisältöön, mutta IEEE:n määrittelemän kehyksen mukaan se sisältää LLC-rakenteen. Kuvassa 5 esitetty LLC-rakenne muodostuu neljästä kentästä, jotka ovat DSAP, SSAP, ohjaus ja data. DSAP ja SSAP ovat lähettäjän ja vastaanottajan palvelupisteet. Niiden tehtävä on sama, kuin Ethernet II -kehysten tyyppi-kentällä, eli eri protokollien erottaminen toisistaan. LLC-rakenteella on yksi poikkeustapaus: mikäli sekä DSAP että SSAP saavat arvon 0xAA, käytetään kuvan 6 mukaista SNAP-kehystä. Se on tavallista LLC-rakennetta pidempi, minkä vuoksi datakentän maksimipituus putoaa hieman. SNAP-kehys koostuu normaalien LLC-kenttien lisäksi organisaatiokohtaisesta SNAP OUI -kentästä ja protokollakohtaisesta SNAP PID -kentästä.

Kehyksen viimeinen kenttä on neljän tavun mittainen tarkiste. Se lasketaan CRC-algoritmilla aloituskenttää lukuun ottamatta koko kehyksestä. Lähetettäessä tarkiste lasketaan ja sijoitetaan lähetettävään kehykseen. Vastaanotettaessa tarkiste lasketaan uudelleen, ja tulosta verrataan kehyksen mukana vastaanotettuun arvoon. Mikäli arvot eivät vastaa toisiaan, kehys hylätään. [6]

3.1.2 MAC-osoitteet

Jokaisella Ethernet-verkon laitteella tulee olla yksilöllinen osoite, joita kutsutaan MAC-osoitteeksi. Osoitteet ovat 48 bitin mittaisia. Osoitteiden kirjoitusasu muodostuu kuudesta heksadesimaaliluvuilla esitetystä tavusta, jotka erotetaan toisistaan kaksoispisteellä. Esimerkiksi pelkistä ykkösistä koostuva yleisjakeluosoite esitetään FF:FF:FF:FF:FF:FF. Kuten yllä todettiin, osoitteen ensimmäinen bitti kertoo, onko vastaanottajana yksittäinen verkon solu, vai onko kyseessä ryhmälähetys. Osoitteen toinen bitti taas kertoo, onko kyseessä aidosti yksilöllinen osoite, vai paikallisesti määritetty osoite. Aidosti yksilölliset osoitteet ovat IEEE:n hallinnoimia. Siltä voi ostaa oikeuden käyttää tiettyä osoiteavaruutta MAC-osoitteissa. Osoitteen ostaminen ei varsinaisesti estä muita valmistajia käyttämästä samaa osoitetta, vaikka se standardien mukaan onkin kiellettyä. Paikallisesti määritettyjä osoitteita hallinnoi paikallisen verkon operaattori.

Aiemmin IEEE myi ainoastaan 22 bittisiä OUI-tunnuksia, mikä jätti ostajalle 2^{24} eli 1 099 511 627 776 osoitetta. Vuodesta 2014 alkaen on siirrytty myymään kolmea erikokoista lohkoa. MA-L vastaa aiemmin myytyä OUI-tunnusta. Sen rinnalla on alettu myymään myös MA-M ja MA-S -lohkoja. MA-M lohkon ostajalle osoitetaan 2^{20} eli 1 048 576 osoitetta, ja MA-S lohkon ostajalle 2^{12} eli 4096 osoitetta. Ainoastaan MA-L lohkon osoitteita voidaan käyttää organisaation tai yrityksen tunnistamiseen.

Yhden MA-L lohkon hinta on kirjoitushetkellä 2575 Yhdysvaltain dollaria [8]. Julkisen MA-L lohkon hinta on kertaluontoinen, eikä erillistä ylläpitomaksua ole. Salaisena pidettävästä tunnisteesta joutuu maksamaan vuosittaisen maksun. [6, 9]

3.1.3 Siirtotien haltuunotto

IEEE:n julkaistessa Ethernet-standardin, se kutsui sitä nimellä CSMA/CD. Termi muodostuu englanninkielisistä sanoista *Carrier Sense, Multiple Access* ja *Collision Detection*. Ne tarkoittavat samassa järjestyksessä kantoaallon tunnistamista, siirtotien jakamista usean laitteen kesken sekä törmäysten tunnistamista. Vuorosuuntaiset Ethernet-verkot käyttävät kyseisen termin kuvaamaa kilpavarausperiaatetta. Se tarkoittaa, että jokainen verkon solu tarkkailee siirtotien tilaa. Mikäli siirtotie on vapaa, mikä tahansa laitteista saa aloittaa lähettämisen. Tämä voi johtaa törmäyksiin, mikäli kaksi solua aloittaa lähettämisen lähes samanaikaisesti. Törmäykset ovat kuitenkin normaali osa verkon toimintaa. Törmäyksen sattuessa molemmat osapuolet havaitsevat sen, ja päättävät toisistaan riippumatta satunnaisesti valitun ajan, jonka jälkeen lähettämistä yritetään uudelleen. Mikäli törmäys tapahtuu uudelleen, kasvatetaan aikaikkunaa, josta viive valitaan. Viesti hylätään vasta kuudentoista peräkkäisen epäonnistuneen lähetyksen jälkeen.

Ethernet-kehyksen minimipituus ja kaapeleiden maksimipituudet ovat seurausta törmäysten tunnistamisesta. Pituudet on määritelty kussakin Ethernetin versiossa siten, että kaikki saman törmäysalueen solut havaitsevat törmäyksen luotettavasti. Teoreettinen kehyksen minimipituus määräytyy siten, että lähetyksen alkupään on ehdittävä kulkea pisimpään kaapelin pituuteen nähden kaksinkertainen matka, ennen kuin lähetys loppuu. Kehyksen minimipituudeksi sovittu 64 tavua ylittää tämän vaatimuksen. Aloituskenttää ei huomioida kehyksen minimipituuteen. [7, 10]

3.2 Fyysinen kerros

IEEE 802.3 määrittelee useita erilaisia fyysisiä kerroksia. Niissä voidaan käyttää kolmea erilaista kaapelityyppiä: koaksiaalikaapelia, valokuitua tai parikaapelia. Koaksiaalikaapeli otettiin käyttöön ensimmäisenä, mutta sitä ei enää suositella uusiin verkkoihin. Valokuitu on teknisesti paras ratkaisu, mutta se on parikaapeliin verrattuna huomattavasti kalliimpaa ja vaikeampaa käsitellä. Parikaapeli onkin edullisuutensa, suorituskykynsä ja helpon asennettavuutensa vuoksi yleisimmin käytetty kaapelityyppi. Eri kaapelityyppien teknisiä ominaisuuksia on verrattu taulukossa 2. Koska muut kaapelityypit ovat selkeästi harvinaisempia, tässä työssä keskitytään parikaapelia käyttäviin Ethernetin versioihin. Kaikki parikaapeliin perustuvat Ethernetin versiot ovat galvaanisesti erotettuja. [4, 10]

EIA/TIA-568-B määrittelee Ethernetissä käytettävät parikaapelikategoriat. Kategorioiden nimet on tapana lyhentää siten, että esimerkiksi kategorian kolme kaapeli merkitään lyhenteellä Cat3. Uusiin asennuksiin suositellaan nopeudesta riippumatta

Taulukko 2: Parikaapeli tarjoaa hyvän suorituskyvyn edulliseen hintaan, ja on siksi yleisimmin käytetty kaapelityyppi. Taulukossa olevat arvot ovat maksimiarvoja. Esimerkiksi valokuidussa maksiminopeus ja kaapelin pituus riippuvat toisistansa. [10]

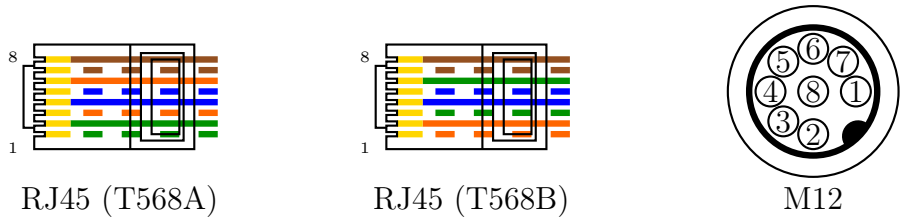
	Parikaapeli	Valokuitu	Koaksiaalikaapeli
Maksiminopeus [Mb/s]	1000	10000	10
Kaapelin maksimipituus	100 m	5 km	500 m
Hinta	Edullinen	Kallis	Keskihintainen
Häiriönsieto	Hyvä	Erinomainen	Hyvä

vähintään Cat5e-luokan kaapeleita. Se on neljästä parista muodostuva balansoitu kaapeli, jossa siirretään differentiaalista signaalia. IEEE 802.3 ei kiellä suojatun kaapelin käyttämistä, mutta yleensä suojaamattoman kaapelin suorituskyky on riittävä, jolloin suojatun kaapelin korkeammalle hinnalle ei saada vastinetta. Suojattua kaapelia käytettäessä sen vaippa tulee maadoittaa vain toisesta päästä, ja lisäksi on huolehdittava impedanssin sovittamisesta. Suojaamattomaan parikaapeliin viitataan lyhenteellä UTP, ja suojattuun lyhenteellä STP. [10]

Parikaapelin johtimet merkitään väreillä taulukon 3 mukaisesti siten, että jokainen pari muodostuu yhdestä värillisestä johtimesta ja sen valkoisesta parista. Valkoinen johdin voidaan merkitä parinsa värin mukaisilla raidoilla. Johtimissa käytettävät värit ovat sininen, oranssi, vihreä ja ruskea. Kaapelin päässä käytetään useimmiten RJ45-liitintä. Sille on olemassa kaksi kytkentätapaa: T568A ja T568B. T568A:n käyttäminen on suositeltavaa. Vaativammassa ympäristössä voidaan käyttää 8-pinnistä M12-liitintä. Tavallisissa kaapeleissa molemmat päät kytketään samalla tavalla. Ris-

Taulukko 3: Cat5-kaapelissa on neljä värikoodattua paria: sininen, oranssi, vihreä ja ruskea. Ne voidaan kytkeä liittimeen joko T568A- tai T568B-standardin mukaisesti. [6]

Pinni	Pari (johtimen väri)	
	T568A	T568B
1	3 (valko-vihreä)	2 (valko-oranssi)
2	3 (vihreä)	2 (oranssi)
3	2 (valko-oranssi)	3 (valko-vihreä)
4	1 (sininen)	1 (sininen)
5	1 (valko-sininen)	1 (valko-sininen)
6	2 (oranssi)	3 (vihreä)
7	4 (valko-ruskea)	4 (valko-ruskea)
8	4 (ruskea)	4 (ruskea)



RJ45 (T568A) RJ45 (T568B) M12

Kuva 7: RJ45-liitin on yleisin parikaapelin kanssa käytettävä liitin. Vaativammissa ympäristöissä voidaan käyttää M12-liitintä. Kuvan RJ45-liittimiin on piirretty taulukon 3 mukaiset kytkentätavat. [10]

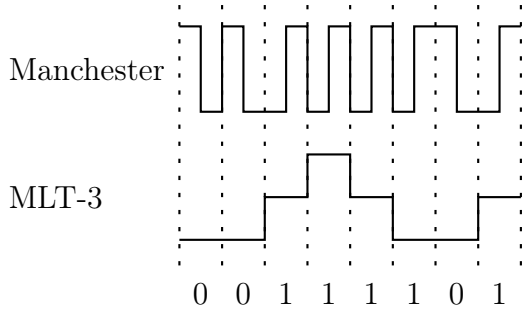
tiinkytketyissä kaapelissa parit kaksi ja kolme kytketään kaapelin toisessa päässä päinvastaiseen järjestykseen. Molemmat liitintyypit ja niiden kytkentätavat on esitetty kuvassa 7. [10]

Useamman solun verkot ovat topologialtaan tähtiä. Niiden keskipisteenä käytetään joko toistinta tai kytkintä. Toistin kytkee eri laitteiden siirtotiet yhteen. Siksi samaan toistimeen kytketyt solut kuuluvat yhteen törmäysalueeseen, jolloin ainoastaan vuorosuuntainen kommunikointi on mahdollista. Toistimeen perustuva verkko muistuttaakin toiminnaltaan enemmän väylää kuin tähteä. Kytkin puolestaan luo jokaiselle verkon solulle oman törmäysalueensa. Se vastaanottaa soluilta saamansa kehykset kokonaan muistiinsa, lukee kehyksestä vastaanottajan osoitteen, ja välittää kehyksen eteenpäin vain halutulle solulle. Siksi kytkin vähentää törmäysten määrää ja sitä kautta nostaa verkon kokonaiskapasiteettia toistimeen verrattuna huomattavasti. Kytkimen avulla voidaan käyttää myös kaksisuuntaista tiedonsiirtoa, jossa törmäyksiä ei tapahdu lainkaan. IEEE 802.3 kuitenkin vaatii, että kaikkien MAC-kerrosten on tuettava myös vuorosuuntaista tiedonsiirtoa. [4, 6]

3.3 Parikaapeliin pohjautuvat versiot

Ethernetistä on yleisimmin käytössä kolme erilaista versiota: 10Base-T, 100Base-TX ja 1000Base-T. Kyseisten termien nimessä oleva numero viittaa suurimpaan mahdolliseen siirtonopeuteen, ja T-kirjain parikaapeliin. Edellä mainittujen lisäksi on olemassa myös vain harvoin käytetyt versiot 100Base-T4 ja 100Base-T2. Niiden tarkoitus oli tukea vanhoja asennuksia, koska 100Base-TX vaatii toimiakseen parempaa kaapelia kuin aiemmin käytössä ollut 10Base-T. Harvinaisuutensa vuoksi ne on rajattu tarkastelun ulkopuolelle. Termi 100Base-T viittaa kaikkiin parikaapelipohjaisiin versioihin, joiden siirtonopeus on 100 Mb/s.

10Base-T on vanhin ja samalla hitain parikaapelia käyttävä Ethernetin versio. Se siirtää tietoa nimensä mukaisesti 10 Mb/s nopeudella ja vaatii toimiakseen vähintään Cat3-tasoisen kaapelin. 10Base-T käyttää ainoastaan pareja kaksi ja kolme: paria kaksi käytetään lähettämiseen ja paria kolme vastaanottamiseen. Tietoa siirretään käyttäen kuvassa 8 esitettyä Manchester-koodausta, jossa nouseva pulssi tarkoittaa binäärilukua 1, ja laskeva pulssi lukua 0. Tahdistamiseen ei tarvita erillistä kellosignaalia, koska jokaisen bitin kohdalla tapahtuu muutos tulkittavassa signaalissa.



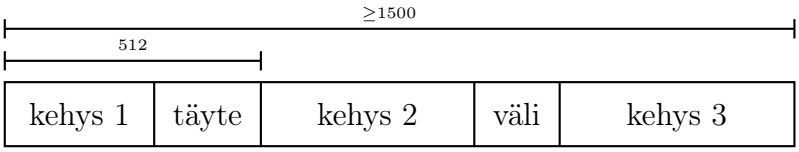
Kuva 8: Manchester-koodauksessa ykkönen esitetään nousevalla reunalla, ja nolla laskevalla reunalla. MLT-3 -koodauksessa käytetään kolmea eri jännitetasoa. Ykkösen kohdalla siirrytään aina seuraavalle jännitetasolle ennalta määrätyssä järjestyksessä. Nollan kohdalla taso ei muutu. [10]

Huonona puolena väylän on pystyttävä vaihtamaan tilaansa kaksinkertaisella nopeudella siirtonopeuteen nähden. Niinpä 10 Mb/s siirtonopeus edellyttää 20 MHz signaalia. [10]

100Base-TX, joka tunnetaan myös englanninkielisellä nimellä *Fast Ethernet*, siirtää tietoa 100 Mb/s nopeudella. Se käyttää siirtotienään vähintään Cat5-tasoista kaapelia, josta 10Base-T:n tapaan käytetään vain pareja kaksi ja kolme. Parempi kaapeli ei kuitenkaan ole ainoa nopeutta lisäävä tekijä. Manchester-koodaus on korvattu kuvassa 8 esitetyllä MLT-3 -koodauksella. MLT-3:ssa data koodataan käyttäen kolmea eri jännitetasoa. Tasot käydään läpi ennalta määrätyssä järjestyksessä siten, että ykkösen kohdalla siirrytään seuraavalle tasolle, ja nollan kohdalla muutosta ei tapahdu. Etuna Manchester-koodaukseen nähden on, että jokaisen siirrettävän bitin kohdalla tarvitaan enintään yksi muutos signaalitasossa, mutta toisaalta runsaasti nollia sisältävään dataan on mahdotonta tahdistua. 100Base-TX ratkaisee tämän ongelman käyttämällä 4B/5B-menetelmää. Ideana on, että jokaiselle 16 erilaisesta neljän bitin kombinaatiosta on määritetty viisibittinen vastine, jossa on tahdistamista varten vähintään kaksi ykköstä. Lopputuloksena 100 Mb/s siirtonopeus vaatii 125 MHz signaalin.

1000Base-T on toistaiseksi nopein käytössä olevista parikaapeliin pohjautuvista Ethernet-verkoista. Se vaatii vähintään Cat5-tasoisen kaapelin ja käyttää kaikkia neljää paria. Linjakoodauksena käytetään PAM5-koodausta. Se hyödyntää viittä eri jännitetasoa, joista jokainen kuvaa kahta bittiä. 1000 Mb/s siirtonopeus saavutetaan, kun kaikkia neljää paria pitkin siirretään kahta bittiä vastaavia tasoja 125 MHz taajuudella. Huomattavaa on, että taajuus on sama, kuin 100Base-TX:llä. Niinpä kaapeleita ei tarvitse uusia, mikäli 100Base-TX verkko halutaan päivittää 1000Base-T:ksi.

Suuremman nopeuden myötä 1000Base-T:n käyttämä kehys poikkeaa aiemmista hieman. Aiemmat versiot määrittelivät kehyksen minimipituudeksi 64 tavua. Suuremman nopeuden myötä kaapelien maksimipituus putoaisi epäkäytännöllisen lyhyeksi. Niinpä 1000Base-T nostaa kehyksen minimipituuden 512 tavuun. Datakentän minimipituus säilyy ennallaan, mutta kehyksen loppuun on lisätty kenttä, jolla minimipituus



Kuva 9: Lyhyitä kehyksiä voidaan 1000Base-T:ssä lähettää purskeena. Ensimmäisen kehyksen on täytettävä yhden kehyksen minimipituus, joka on 512 tavua. Koko lähetyksen minimipituus on 1500 tavua. [7]

tarvittaessa saavutetaan. Tämä ylimääräinen kenttä ei sisällä ylemmille kerroksille välitettävää tietoa. Lisäksi kehyksiä voidaan lähettää purskeena kuvan 9 mukaisesti. Ensimmäisen kehyksen on ylitettävä 512 tavun minimipituus, minkä jälkeen kehyksiä lähetetään niin monta, että lähetyksen yhteispituus on vähintään 1500 tavua. Menetelmä nostaa lyhyiden kehysten hyötysuhdetta merkittävästi. [7]

Ethernetin eri versiot eivät ole suoraan yhteensopivia toistensa kanssa, vaikka fyysinen liitäntä onkin yhtenevä. Useampia erityyppisiä versioita tukevat laitteet voivat kuitenkin käyttää *Auto-Negotiation* -termillä tunnettua menetelmää, joka vapaasti suomennettuna tarkoittaa automaattista neuvottelua tai selvittämistä. Kyseessä on menetelmä, jolla kaksi laitetta voi sopia keskenään käyttämistään siirtotien parametreista, kuten verkon nopeudesta ja kaksisuuntaisuudesta. Auto-Negotiation on käytettävissä lähes kaikissa moderneissa Ethernet-laitteissa. [6, 11]

3.4 Tyypilliset komponentit

Tyypillinen Ethernet-liityntä muodostuu neljästä eri komponentista: MAC-piiristä, PHY-piiristä, edellä mainittujen piirien rajapinnasta sekä erotusmuuntajasta. MAC-piiri vastaa MAC-tason toteutuksesta. Piirien rajapinta on usein toteutettu MAC-piirissä, mutta myös ulkoisia toteutuksia on olemassa. PHY-piiri puolestaan vastaa yhdessä erotusmuuntajan kanssa fyysisen kerroksen toteutuksesta. Erotusmuuntajan tyypillinen jännitekesto on 1500 VDC. Teolliseen käyttöön on kuitenkin saatavilla mitoitukseltaan parempia komponentteja, esimerkiksi suurempia jännitteitä kestäviä erotusmuuntajia ja yhteismuotoista signaalia paremmin vaimentavia komponentteja. Edellä mainittujen komponenttien lisäksi Ethernet-laitteissa on usein kaksi merkkivaloa. Niistä toinen kertoo yhteyden olevan toiminnassa, ja toinen ilmaisee lähetyksen olevan käynnissä. [11]

IEEE:n määrittelemä MAC- ja PHY-piirien rajapinta riippuu käytettävästä siirtonopeudesta. Ensimmäisenä tarkasteltu 10 Mb/s Ethernet kutsuu tätä rajapintaa AUI:ksi. Nimitys muodostuu englanninkielisistä sanoista Attachment Unit Interface, tarkoittaen liitäntäyksikön rajapintaa. 100 Mb/s Ethernet kutsuu vastaavaa rajapintaa nimellä MII. Se puolestaan muodostuu englanninkielisistä sanoista *Media Independent Interface* ja tarkoittaa siirtotiestä riippumatonta rajapintaa. 1000 Mb/s Ethernet päivitti edelleen saman rajapinnan nimeksi *Gigabit MII* eli GMII, jossa G-kirjain viittaa 1000 Mb/s siirtonopeuteen. MII ja GMII ovat taaksepäin yhteensopivia hitaampien edeltäjiensä kanssa. [11, 12]

Taulukko 4: MII-rajapinta muodostuu 16 signaalista. [12]

Signaalin nimi	Merkitys
RX_CLK	Vastaanottamisen kellosignaali
RXD	Vastaanottamisen datasignaalit (4 kpl)
RX_DV	Vastaanottamisen erotin
RX_ER	Vastaanottamisen virhesignaali
TX_CLK	Lähettämisen kellosignaali
TXD	Lähettämisen datasignaalit (4 kpl)
TX_EN	Lähettämisen erotin
TX_ER	Lähettämisen virhesignaali
CRS	Siirtotien vapaus
COL	Törmäys

MII muodostuu yhteensä 16 signaalista, jotka on esitetty taulukossa 4. Molempiin suuntiin kulkee neljän bitin dataväylä, erotinsignaali, virhesignaali ja kellosignaali. Jäljelle jäävät kaksi signaalia ovat siirtotien vapaudesta kertova CRS-signaali ja törmäyksen tapahtumisesta kertova COL-signaali. MII tukee sekä 10 Mb/s että 100 Mb/s nopeutta, niiden ainoa ero on tiedonsiirron nopeus. 100 Mb/s nopeudella jokainen neljästä dataväylästä siirtää tietoa 25 MHz taajuudella. GMII nostaa dataväylien määrän neljästä kahdeksaan suuntaa kohden, jolloin se muodostuu yhteensä 24 signaalista. Lisäksi toimintataajuutta on nostettu, 125 MHz signaali kahdeksassa dataväylässä tuottaa 1000 Mb/s nopeuden. Samalle MII-väylälle on mahdollista kytkeä useita PHY-piirejä, kunhan niille on määritetty yksilölliset PHY-osoitteet. Vaikka useamman PHY-piirin käyttäminen ei tässä työssä ole tarpeen, on tunnettava osoitteen merkitys, jotta piirien välinen kommunikointi toimisi. Vain yhtä PHY-piiriä käytettäessä sen tulee vastata osoitetta 0. [12]

Kaikki edellä mainitut rajapinnat ovat IEEE:n määrittelemiä. Niiden lisäksi on olemassa myös pelkistetty rajapinta, joka tunnetaan sen englanninkielisellä nimellä

Taulukko 5: RMII-rajapinta vähentää tarvittavien signaalien määrän puoleen, muodostuen kahdeksasta signaalista. [13]

Signaalin nimi	Merkitys
RXD	Vastaanottamisen datasignaalit (2 kpl)
CRS_DV	Yhdistetty CRS ja RX_DV
RX_ER	Vastaanottamisen virhesignaali
TXD	Lähettämisen datasignaalit (2 kpl)
TX_EN	Lähettämisen erotin
REF_CLK	Yhteinen kellosignaali

Reduced MII eli RMII. Se on RMII Consortiumin kehittämä rajapinta, jonka tarkoitus on luoda edullisempi vaihtoehto IEEE:n määrittelemälle MII-rajapinnalle vähentämällä MAC- ja PHY-piirien välillä tarvittavien signaalin määrää. Se tukee MII:n tavoin sekä 10 Mb/s että 100 Mb/s nopeutta. RMII vaatii ainoastaan kahdeksan signaalia, mikä on puolet MII:n signaaleista. Tätä varten rajapinnan kellotaajuus on nostettu 25 MHz:sta 50 MHz:iin, jolloin datasignaalien määrä voidaan puolittaa. COL-signaali on jätetty pois, CRS on yhdistetty RX_DV:hen, ja lähettämisen ja vastaanottamisen kellosignaalit on korvattu yhteisellä kellosignaalilla. RMII:n signaalit on koottu taulukkoon 5. [13]

PHY-piirien sisältämät rekisterit on yhteensopivuuden takaamiseksi standardoitu. Rekistereitä on vähintään kaksi, ja enintään 32. Kukin rekistereistä on 16 bitin kokoinen. MII ja GMII määrittelevät kumpikin kaksi erilaista rekisterien kokonaisuutta: perustason, ja laajennetun. Mikäli PHY-piiri tukee MII- tai GMII-rajapintaa, sen on toteutettava perustason rekisterit. Laajennettua rekisterikokonaisuutta ei ole pakko toteuttaa, mutta toteutetut rekisterit eivät saa myöskään olla ristiriidassa sen kanssa. Rekisterit 16...31 ovat vapaasti piirin valmistajan käytettävissä. Taulukossa 6 on esitetty rekisterien yhteenveto. [12]

Taulukko 6: MII- ja GMII-rajapinnat standardoivat PHY-piirin rekisterit. B-kirjaimella merkityt rekisterit kuuluvat perustason rekistereihin, ja E-kirjaimella merkityt laajennettuihin rekistereihin. [12]

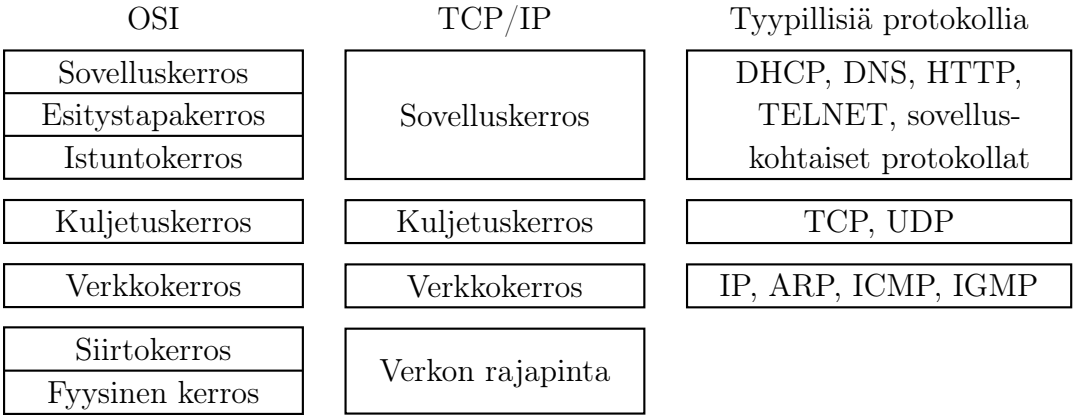
Osoite	Rekisterin nimi	MII	GMII
0	Control	B	B
1	Status	B	B
2, 3	PHY identifier	E	E
4	Auto-Negotiation Advertisement	E	E
5	Auto-Negotiation Link Partner Base Page Ability	E	E
6	Auto-Negotiation Expansion	E	E
7	Auto-Negotiation Next Page Transmit	E	E
8	Auto-Negotiation Link Partner Received Next Page	E	E
9	MASTER-SLAVE Control Register	E	E
10	MASTER-SLAVE Status Register	E	E
11	PSE Control Register	E	E
12	PSE Status Register	E	E
13	MMD Access Control Register	E	E
14	MMD Access Address Data Register	E	E
15	Extended Status	-	B
16...31	Valmistajakohtaisia rekistereitä	E	E

4 TCP/IP-protokollaperhe

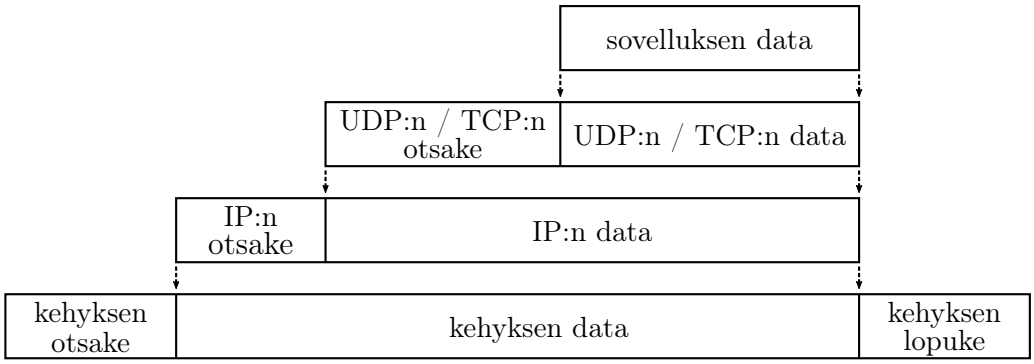
OSI-viitekehyksen alimmille tasoille sijoittuva Ethernet vastaa siitä, että verkon laitteet voivat lähettää kehyksiä toisillensa. Se ei kuitenkaan tarkoita, että ne ymmärtäisivät toisiansa. Tilannetta voi verrata ihmisiin, jotka eivät puhu samaa kieltä. Molemmat kommunikoivat ääniaaltojen välityksellä, mutta yhteisen kielen puuttuessa kommunikointi ei ole mahdollista. Tässä luvussa tutustutaan Ethernetin päällä käytettävään TCP/IP-protokollaperheeseen. [5]

TCP/IP kehitettiin Yhdysvaltain puolustusministeriössä 1970-luvun taitteessa. Sitä käytettiin alun perin hallituksen, sotilaallisten organisaatioiden ja koulutuslaitosten välisenä verkkona, mutta myöhemmin TCP/IP avattiin myös kaupalliseen käyttöön. Nykyään se on käytössä maailmanlaajuisesti, sillä koko Internet rakentuu TCP/IP:n varaan. TCP/IP ei ole yksittäinen protokolla, vaan se kokoelma erilaisia protokollia. Sen on nimetty kahden tärkeimmän protokollansa, TCP:n ja IP:n, mukaan. Koko TCP/IP-protokollaperheen käsitteleminen ei ole tämän työn puitteissa tarkoituksenmukaista, eikä edes mahdollista. Käsittelyyn on valittu joitakin keskeisimmistä protokollista.

Mikäli TCP/IP:tä verrataan OSI-viitekehykseen, se sijoittuu tasoille 3-7. TCP/IP määrittelee kuitenkin oman viitekehyksensä, joka on esitetty kuvassa 10. Alimmalla tasolla oleva verkon rajapinta on tämän työn tapauksessa Ethernet, mutta TCP/IP:tä käytetään monissa muissakin verkoissa. Verkkokerros ja kuljetuskerros vastaavat OSI:n mukaisia kerroksia, mutta sitä korkeammat tasot on yhdistetty yhdeksi kerrokseksi, jota kutsutaan sovelluskerrokseksi. Sovelluskerroksen data kapseloidaan kuvan 11 mukaisesti joko UDP-sähkeeseen tai TCP-segmenttiin ja edelleen IP-pakettiin. IP-paketti kapseloidaan edelleen verkon rajapinnan mukaiseen kehykseen, joka tämän työn tapauksessa on Ethernetin määrittelemä MAC-kehys. [6]



Kuva 10: TCP/IP:n viitekehys muistuttaa OSI-viitekehystä, mutta vähentää tasojen määrän seitsemästä neljään. TCP/IP ei ota kantaa alemman tason ratkaisuihin, joten protokollaa voidaan käyttää monissa erityyppisissä verkoissa. Kuvaan merkityt protokollat ovat esimerkkejä tyypillisistä protokollista. [6]



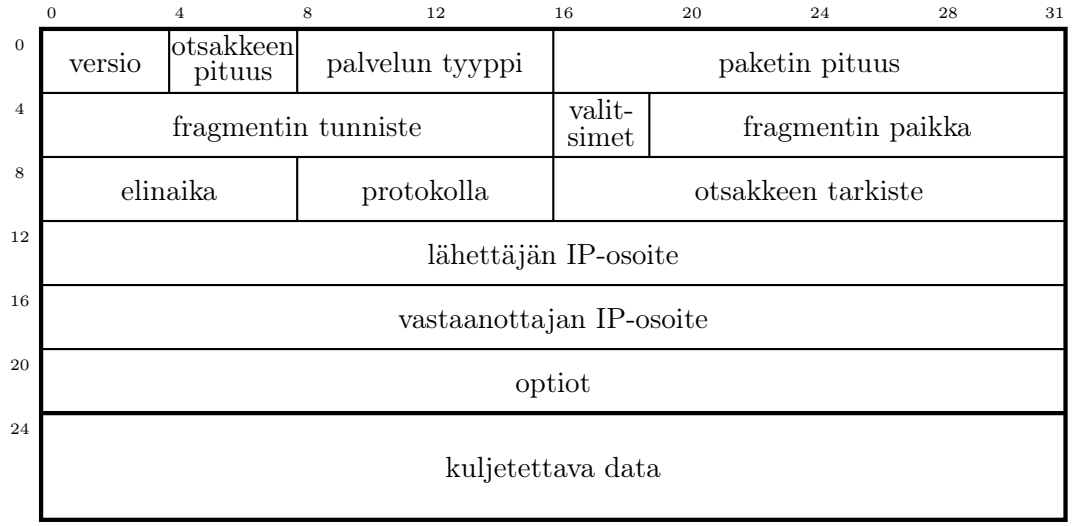
Kuva 11: Sovelluksen data kapseloidaan TCP/IP-perheessä joko UDP-sähkeeseen tai TCP-segmenttiin. Kuvaan merkitty alimman tason kehys on Ethernetin tapauksessa MAC-kehys. [6]

4.1 IP-protokolla

Verkkokerroksen keskeisin protokolla on IP. Se käy parhaillaan läpi murrosta, sillä alkuperäisen IPv4-protokollan osoiteavaruus on loppumassa kesken. Siirtymävaihe IPv6-protokollaan on jo alkanut, mutta IPv4:n käyttämiselle ei ole määritelty takarajaa, ja IPv4 on yhä yleisimmin käytetty versio. Tässä luvussa käsitellään IPv4:n mukaisia protokollia, ja lopuksi tarkastellaan lyhyesti, miten IPv6 eroaa IPv4:stä. [5]

IPv4 määrittelee kuvassa 12 esitetyn paketin. Se muodostuu 13 kentän mittaisesta otsakkeesta ja kuljetettavasta datasta. Otsakkeen kentät ovat versio, otsakkeen pituus, palvelun tyyppi, paketin pituus, fragmentin tunniste, valitsimet, fragmentin paikka, elinaika, protokolla, otsakkeen tarkiste, lähettäjän ja vastaanottajan IP-osoitteet, sekä optiot. Versio on ainoa yhteinen kenttä IPv4:n ja IPv6:n välillä. Se on kullekin protokollan versiolle vakio, joten eri version mukaiset paketit on helppo erottaa toisistaan. Otsakkeen pituus määrittää otsakkeen pituuden neljän oktetin kerrannaisina. Pituus ilmoitetaan neljällä bitillä ja se määritellään neljän oktetin kerrannaisina, mikä rajoittaa otsakkeen maksimipituudeksi 60 oktettia. Palvelun tyyppi -kenttä on tarkoitettu pakettien priorisointia varten. Sitä käytetään sovelluksissa, jotka vaativat reaaliaikaista dataa. Äänivirta on tyypillinen esimerkki tällaisesta sovelluksesta.

Paketin pituus määrittää koko paketin pituuden tavuina. Teoreettinen maksimi on kentän koon myötä 2^{16} tavua, mutta Ethernetissä tavanomainen rajoite on paketin pirstoutumisen välttämiseksi 1500 tavua. Mikäli paketti on liian suuri, se pilkotaan lähetettäessä pienemmiksi fragmenteiksi. Fragmentin tunniste, valitsimet ja fragmentin paikka ovat käytössä, kun pilkottua pakettia kootaan uudelleen vastaanottavassa laitteessa. Tunnisteen avulla erotetaan samaan pakettiin kuuluvat fragmentit muista paketeista, ja fragmentit järjestetään oikeaan järjestykseen fragmentin paikka -kentän avulla. Valitsimet-kenttä koostuu kolmesta bitistä, joista ensimmäinen on aina ykkönen. Paketin pirstoutuminen voidaan kieltää asettamalla toinenkin bitti ykköseksi. Tämä voi kuitenkin johtaa pakettien hylkäämiseen, mikäli ne ovat liian suuria jollekin verkon osalle. Kolmas bitti määrittää, onko kyseessä paketin viimeinen fragmentti. Kolmas bitti on aina nolla niissä paketeissa, joita ei pilkota fragmenteiksi.



Kuva 12: IPv4-protokollan mukainen paketti. Paketin otsake muodostuu 13 kentästä. Vaakasuunnan numerot ovat bittien, ja pystysuunnan numerot oktettien järjestysnumeroita. [5]

Paketille päätetään elinaika ennen sen lähettämistä. Jokainen paketteja välittävä laite pienentää kentän arvoa. Mikäli se saavuttaa nollan ennen kuin paketti saavuttaa vastaanottajan, paketti hylätään ja lähettäjä saa ICMP-protokollan kautta ilmoituksen hylätystä paketista. Elinajan tarkoitus on estää pakettien jumiutuminen verkkoon. IP:n sisälle kapseloidun paketin tyyppi määritetään protokolla-kentällä. IANA on standardoinut protokollia vastaavat numerot. Taulukkoon 7 on koottu joitakin yleisimpiä protokollia vastaavat numerot. Täydellinen listaus on saatavilla IANA:n verkkosivujen [14] kautta. Otsakkeen tarkiste sisältää 32-bittisen tarkisteen koko muusta otsakkeesta. Tarkiste voi muuttua paketin kulkiessa verkon lävitse, sillä esimerkiksi paketin elinaika muuttuu reitittimien kohdalla. Otsakkeen viimeiset kentät ovat lähettäjän ja vastaanottajan IP-osoitteet sekä optiot. IP-osoitteita käsitellään tarkemmin luvussa 4.2. Optiot-kenttä ei ole pakollinen, eikä sitä yleensä käytetä. Optioiden avulla voidaan esimerkiksi tallettaa paketin kulkema reitti. Niiden käyttöön kuitenkin liittyy myös ongelmia, ja sen vuoksi jotkin operaattorit eivät hyväksy optioiden käyttämistä lainkaan. [5, 6]

Taulukko 7: Luettelo IPv4:n yleisimpiä protokollia vastaavista numeroista. Luvut on esitetty sekä 10-kantaisina että 16-kantaisina. [5]

Protokolla	10-kantainen arvo	16-kantainen arvo
ICMP	1	0x01
IGMP	2	0x02
TCP	6	0x06
UDP	17	0x11

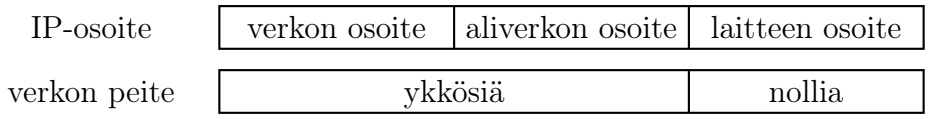
4.2 IP-osoitteet ja verkon peite

Jokaiselle verkon laitteelle osoitetaan yksilöllinen IP-osoite, jota ei MAC-osoitteen tapaan sidota tiettyyn laitteeseen. Osoitteet ovat käytännössä 32-bittisiä lukuja, mutta selkeyden vuoksi ne on tapana esittää neljänä yhden oktetin lukuna. Esimerkiksi binääriluku 11000000 10101000 0000001 00000001 vastaa IP-osoitetta 192.168.1.1. Osoite voidaan myös esittää heksadesimaalilukujen avulla, jolloin edellä mainittu osoite olisi C0.A8.01.01. Verkon peite liittyy läheisesti IP-osoitteisiin. Peite on 32 bitin kuvio, jolla rajoitetaan verkon kokoa. Kuvion alkuosa muodostuu pelkistä ykkösistä ja loppuosa pelkistä nollista. Ykkösten rajoittamat bitit ovat saman verkon IP-osoitteissa aina samat, ja nollien rajaama alue on laitekohtainen. Esimerkiksi peite 11111111 11111111 11111111 00000000 tarkoittaisi edellä käytetyssä esimerkissä, että kaikkien verkon laitteiden IP-osoite alkaisi 192.168.1. Nollaan päättyvät osoitteet ovat verkon osoitteita ja numeroon 255 päättyvät osoitteet monilähetysosoitteita, joten esimerkin verkossa voisi olla enintään 254 laitetta. Tämä osoitealue voidaan esittää muodossa 192.168.1.0/24, jossa luku 24 viittaa verkon peitteessä olevien ykkösten määrään. [5]

Alun perin IP-osoitteet jakautuivat kahteen kenttään, jotka olivat verkon tunniste ja laitteen tunniste. Osoitteet oli jaettu viiteen eri luokkaan, joita kuvattiin kirjaimilla A...E. Näistä olennaisimpia olivat luokat A, B ja C, joiden rakenne on esitetty kuvassa 13. A-luokan osoitteissa verkon tunniste koostui seitsemästä bitistä, ja laitteen tunniste 24 bitistä. B-luokan osoitteissa verkon tunnisteelle oli varattu 14 bittiä, ja laitteen tunnisteelle 16 bittiä. C-luokassa verkon tunniste oli 20-bittinen, ja laitteen tunniste 8-bittinen. D-luokan osoitteet oli varattu monilähetyksille, eikä niitä sen vuoksi jaettu erikseen verkon ja laitteen tunnisteisiin. E-luokan osoitteet oli varattu myöhempää käyttöä varten. Jokaiselle Internetiin liittyvälle verkolle osoitettiin oma verkon tunniste sellaisesta luokasta, joka takasi riittävän määrän laitteen tunnisteita.

IP-osoitejärjestelmää uudistettiin vuonna 1993, koska osoitteiden jako kolmeen pääluokkaan ei ollut osoitteiden kulutuksen kannalta tehokasta. Niinpä luokkajaosta luovuttiin, mutta luokkia vastaavat termit jäivät yhä yleiseen käyttöön. Osoitteiden

Kuva 13: IP-osoitteet jakautuivat aluksi viiteen eri luokkaan. Luokkien A...C osoitteet ovat tavallisia IP-osoitteita. D-luokan osoitteita käytettiin monilähetyksissä ja E-luokan osoitteet oli varattu myöhempää käyttöä varten. [7]



Kuva 14: IP-osoite muodostuu kolmesta osasta, jotka ovat verkon osoite, aliverkon osoite ja laitteen osoite. Osoitetta vastaava verkon peite kattaa verkon ja aliverkon osoitteet. [5]

koko on uudistuksen jälkeen ennallaan, mutta ne muodostuvat kuvan 14 mukaisesti kolmesta kentästä, jotka ovat verkon osoite, aliverkon osoite ja laitteen osoite. Osoitetta vastaava verkon peite kattaa kuvan mukaisesti verkon ja aliverkon osoitteet. Kenttien kokoja ei ole lukittu, mutta tyypillisesti paikallisissa verkoissa käytetään C-luokan osoitetta, jossa verkon osoite on 16-bittinen, ja aliverkon sekä laitteen osoitteet 8-bittisiä. Koska kenttien kokoja ei ole lukittu, jokaiselle verkolle voidaan osoittaa sopivan kokoinen osoiteavaruus.

Mikäli verkko ei ole missään yhteydessä Internetiin, sen IP-osoitteet voidaan periaatteessa määrittää mielivaltaisesti. On kuitenkin erittäin suositeltavaa noudattaa IANA:n mukaisia osoitteita kaikissa verkoissa, vaikka ne eivät olisikaan yhteydessä Internetiin. IANA on varannut paikallisille verkoille lohkon A-, B- ja C-luokista. IANA:n määrittämät lohkot on esitetty taulukossa 8. Näiden lisäksi 169.254.0.0/16 -avaruus on varattu paikallisten verkkojen sisäiseen kommunikointiin. Kyseisen lohkon englanninkielinen nimi on *link-local block*. [5, 7]

Laitteen IP-osoite voidaan määrittää kahdella eri tavalla: joko staattisesti tai dynaamisesti. Staattiset osoitteet määritetään laitteelle sen käyttöönoton yhteydessä. Staattinen osoite voidaan tarvittaessa vaihtaa, mutta tyypillisesti se on pysyvä. Dynaaminen osoite taas määritetään aina, kun laite liittyy verkkoon. Tällöin osoite määritetään yleensä luvussa 4.7 käsiteltävän DHCP-protokollan avulla. Staattisten osoitteiden etuna on se, että ne pysyvät vakioina, eivätkä vaadi palvelinta osoitteiden määrittämiseen. Dynaamisten osoitteiden etu taas on se, että osoitteet määrittyvät automaattisesti, eivätkä verkon laitteet vaadi erillistä konfigurointia. [5]

Taulukko 8: IANA:n varaamat osoiteavaruudet paikallisille verkoille, joista on yhteys Internetiin. [5]

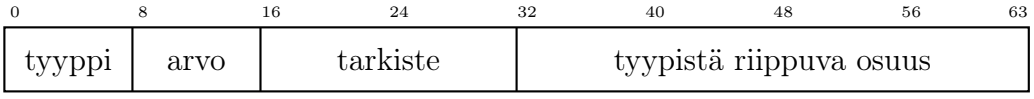
Luokka	Osoiteavaruus	Mahdolliset osoitteet		
A	10.0.0.0/8	10.0.0.0	...	10.255.255.255
B	172.16.0.0/12	172.16.0.0	...	172.31.255.255
C	192.168.0.0/16	192.168.0.0	...	192.168.255.255

4.3 Muut verkkokerroksen protokollat

Verkkokerrokseen kuuluu IP-protokollan lisäksi muitakin protokollia. IP on kuitenkin ainoa verkkokerroksen protokolla, joka kuljettaa sovellustason dataa laitteiden välillä. Muut protokollat tukevat IP-protokollan toimintaa esimerkiksi selvittämällä osoitteita sekä raportoimalla epänormaaleista tilanteista. Muiden protokollien viestit kapseloidaan joko IP-paketteihin tai suoraan alemman tason kehyksiin. Tässä luvussa käsiteltävät protokollat ovat ARP, ICMP ja IGMP. Koska IP on verkkokerroksen tärkein protokolla, muita protokollia ei käsitellä yhtä yksityiskohtaisesti.

Kun verkossa oleva laite on lähettämässä IP-pakettia toiselle laitteelle, sen on tiedettävä vastaanottajan IP-osoite. Tämä ei kuitenkaan riitä, koska alemman tason protokollat tarvitsevat kehyksen lähettämiseen myös alemman tason osoitteen. Ethernetin tapauksessa tämä tarkoittaa MAC-osoitteita. ARP, joka on englanninkieliseltä nimeltään *Address Resolution Protocol*, on protokolla, jonka avulla selvitetään IP-osoitetta vastaava alemman tason osoite. Osoite selvitetään lähettämällä verkon yleisjakeluosoitteeseen kysely halutun IP-osoitteen haltijasta. Kyselyyn liitetään lähettäjän tiedot, jolloin selvitettävän osoitteen haltija tietää, mihin osoitteeseen vastaus tulee lähettää. Osoitteet tallennetaan ARP-taulukkoon kyselyjen vähentämiseksi. Mikäli osoitetta ei käytetä uudelleen ennalta määrätyssä aikaikkunassa, se poistetaan taulukosta. Aikaikkunan pituus on toteutuskohtainen, mutta tyypillisesti se on muutaman minuutin luokkaa.

ICMP, eli *Internet Control Message Protocol*, on protokolla jota käytetään verkon poikkeustilanteissa. ICMP:n viestejä kuljetetaan IP-protokollan avulla. Ne jakautuvat kahteen pääryhmään: tiedusteluihin, ja virheilmoituksiin. Kuvassa 15 esitetty ICMP:n otsake kapseloidaan IP-paketin datakenttään. Viestin ensimmäinen tavu kertoo viestin tyypin. Esimerkiksi arvoa kolme vastaa ICMP-viesti, joka kertoo, että aiemmin lähetetyn IP-paketin vastaanottajaa ei tavoitettu. Viestin toinen tavu on kullekin viestityypille ominainen arvo tai parametri, joka voi esimerkiksi määritellä virhetilanteen syyn. Tyypin ja arvon jälkeen otsakkeessa on kahden tavun kokoinen tarkiste, ja neljä tavua viestin tyypistä riippuvaa dataa. Näiden lisäksi kehykseen voi kuulua myös erillinen datakenttä. *Echo*, eli kaiku, on tyypillinen esimerkki tiedustelusta. Kyseessä on ICMP-viesti, jolla testataan yhteyttä johonkin verkon soluun. Echo-viestin vastaanottanut solu kuittaa viestin, jolloin lähettäjä tietää yhteyden toimivan. Menetelmä tunnetaan myös ping-ohjelmana, joka sisältyy kaikkiin TCP/IP:tä tukeviin käyttöjärjestelmiin. ICMP koostuu yhteensä 15 erityyppisestä viestistä, mutta niiden tukeminen on sovelluskohtaista. Lisäksi reitittimet voivat hylätä joitakin ICMP-viestejä turvallisuussyistä. [5, 7]



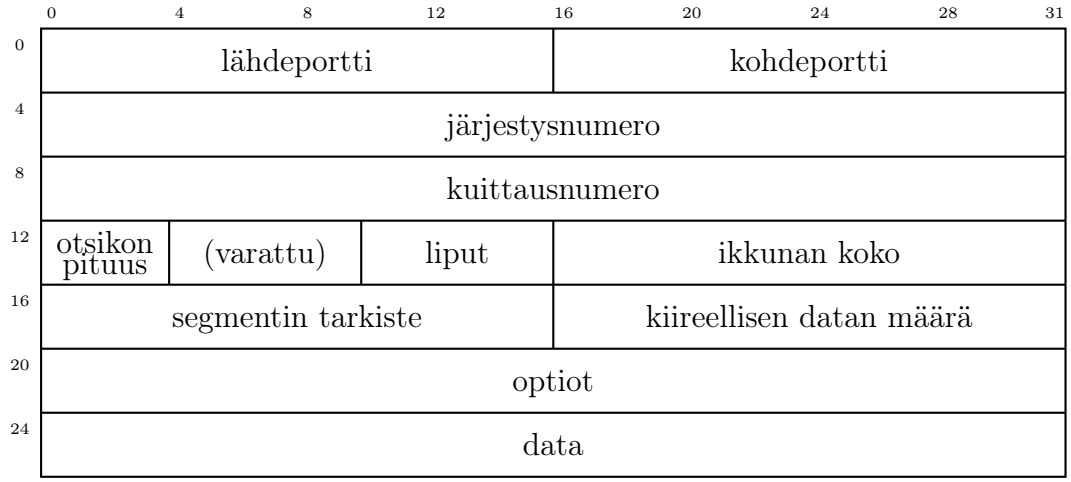
Kuva 15: ICMP-protokollan otsake. Kehykseen voi kuulua myös datakenttä. [5]

IGMP, joka on lyhenne englanninkielisistä sanoista *Internet Group Management Protocol*, on datavirtojen monilähetyksissä käytettävä protokolla. Se muistuttaa edellisessä kappaleessa käsiteltyä ICMP:tä, sillä molempien viestit kulkevat IP-paketeissa. IGMP:n paketteja ei kuitenkaan osoiteta yksittäiselle vastaanottajalle, vaan erityiselle monilähetykseen käytettävälle osoitteelle. Laitteen, joka tahtoo vastaanottaa IGMP-datavirtaa, on kuunneltava oman IP-osoitteensa lisäksi kyseisen IGMP:n monilähetysosoitetta. IGMP:tä voidaan käyttää esimerkiksi ääni- ja videovirtojen lähettämiseen. [15]

4.4 TCP-protokolla

Transmission Control Protocol, eli TCP, on sovelluskerroksen tärkein yksittäinen protokolla. IP-protokollan kuljettaessa dataa kahden laitteen välillä, TCP vastaa sovellusten välisestä kommunikoinnista. Se on yhteydellinen protokolla, mikä tarkoittaa, että ennen tiedonsiirtoa yhteys on avattava. Yhteys muodostuu lähteestä ja kohteesta, joille molemmille määritetään yhteyttä varten portti. Portti on käytännössä 16-bittinen numero, jonka mahdolliset arvot ovat välillä 0 ... 65535. Portin ei tarvitse olla yhteyden eri päissä sama. Portteja käsitellään tarkemmin luvussa 4.6. TCP:tä käyttävä sovellus voi luottaa tiedonsiirron tekniseen virheettömyyteen, mutta tietoturvan kannalta se on suojaamaton.

TCP-protokolla kuljettaa dataa segmenteissä, jotka kapseloidaan IP-paketteihin. Yksittäisen segmentin rakenne on esitetty kuvassa 16. Se koostuu lähde- ja kohdeporteista, järjestys- ja kuittausnumeroista, otsikon pituudesta, lipuista, ikkunaan koosta, segmentin tarkisteesta, kiireellisen datan määrästä, optioista sekä datasta. Liput-kenttä muodostuu kuudesta bitistä, jotka ovat URG, ACK, PSH, RST, SYN ja FIN. Bitit lyhennetään toisinaan alkukirjaimen mukaan, jolloin esimerkiksi U-bitti tarkoittaa URG-bittiä. Kunkin bitin merkitys on kuvattu taulukossa 9. Yhteys



Kuva 16: TCP-segmentin rakenne. Otsikon pituuden ja lippujen väliin jäävä kuuden bitin alue on varattu myöhempää käyttöä varten. [5]

Taulukko 9: TCP-segmentin liput-kentän bittien kuvaus. [4]

Lippu	Merkitys
URG	Ilmoittaa, että segmentin datakentässä on kiireellistä dataa
ACK	Segmentti kuittaa aiemmin lähetetyn segmentin
PSH	Siirretty data tulee luovuttaa ylemmälle kerrokselle heti
RST	Keskeyttää yhteyden tai sen muodostamisen
SYN	Segmentti, joka synkronoi järjestysnumerot yhteyttä avattaessa
FIN	Yhteyden lopetuspyyntö

muodostuu aina kahden laitteen välille, eikä monilähetys ole TCP:ssä mahdollista. Yhteyden avaajaa kutsutaan asiakkaaksi ja toista osapuolta palvelimeksi. Yhteys avataan kolmivaiheisella kättelyllä alla kuvatulla tavalla.

1. Asiakas lähettää avaussegmentin palvelimelle, ennalta määrättyyn porttiin. Avaussegmentti määrittelee asiakkaan käyttämän portin, järjestysnumeroiden alkuarvon ja suurimman segmentin koon, joka voidaan siirtää ilman pirstaloitumista. Asiakkaan portti määräytyy lähdeportista. Ensimmäinen järjestysnumero on asiakkaan satunnaisesti valitsema 32-bittinen luku, joka synkronoidaan palvelimelle asettamalla SYN-lippu päälle. Segmentin järjestysnumero ei kasva segmenttien määrän, vaan siirrettyjen dataoktettien mukaan. Suurimman segmentin koko välitetään otsikon optioissa.

2. Palvelin kuittaa vastaanottamansa segmentin. Segmentin ACK-lippu on aktivoitu merkiksi kuittauksesta. Kuittaavan paketin kuittausnumero kertoo järjestysnumeron seuraavalle lähetettävälle dataoktetille. Koska eri suuntiin kulkevalle datalle käytetään toisistaan riippumattomia segmentin numeroita, myös palvelin valitsee satunnaisen numeron ja synkronoi sen asiakkaalle SYN-lipun avulla. Lisäksi palvelinkin kertoo suurimman, ilman pirstaloitumista siirrettävissä olevan segmentin koon.

3. Asiakas kuittaa vastaanottaneensa palvelimen paketin, jolloin yhteys katsotaan avatuksi.

Kun yhteys on avattu, molemmat osapuolet voivat lähettää segmenttejä sen yli. Jokaiselle segmentille annetaan järjestysnumero, jota kasvatetaan aina siirrettyjen dataoktettien määrän mukaan. Vastaanottaja kuittaa jokaisen vastaanotetun segmentin samaan tapaan kuin yhteyttä avattaessa. Lähettäjä ei lähetä uutta segmenttiä ennen kuin edellinen segmentti on kuitattu. Mikäli segmenttiä ei kuitata määräajassa, se lähetetään uudelleen. Ikkunan koko kertoo, kuinka suuren datamäärän lähettäjä on valmis vastaanottamaan. [5, 15]

Segmentille voidaan määrittää lippujen avulla joitakin normaalista poikkeavia toimintoja. URG-lipulla merkityt fragmentit sisältävät kiireellistä dataa. Kiireellinen

data sijoitetaan datakentän alkuun ja sen määrä päätellään otsikon kiireellisen datan määrä -kentästä. Kiireellinen data voidaan käsitellä muusta datasta erillään ja sen tehtävä määritellään sovelluskohtaisesti. PSH-lipulla merkitty segmentti taas tarkoittaa, että vastaanotettu datavirta tulee luovuttaa siitä vastaavalle sovellukselle. Mikäli P-lippua ei ole asetettu, dataa voidaan puskuroida, jolloin sovellukselle luovutetaan useamman segmentin data kerralla. Kolmas erikoistilanne on yhteyden keskeyttäminen, joka tehdään RST-lipun avulla. Keskeyttäminen ei ole normaali tapa sulkea yhteys, vaan sitä käytetään, kun jompikumpi osapuoli havaitsee, ettei yhteys enää toimi normaalisti. [5, 7]

Yhteyden sulkeminen tapahtuu neljän alla kuvatun segmentin avulla. Kumpi tahansa osapuoli voi aloittaa yhteyden sulkemisen.

1. Aloitteen tekevä osapuoli lähettää segmentin, jossa FIN-lippu on aktivoitu. Se merkitsee, että osapuoli ei enää lähetä uutta dataa.

2. Toinen osapuoli kuittaa segmentin, jolloin sillä on kaksi vaihtoehtoa. Tiedonsiirto voi vielä jatkua yksisuuntaisena, mikäli toisella osapuolella on dataa, jonka se tahtoo lähettää ennen yhteyden sulkemista. Mikäli tiedonsiirtoa jatketaan, segmentit kuitataan normaaliin tapaan,

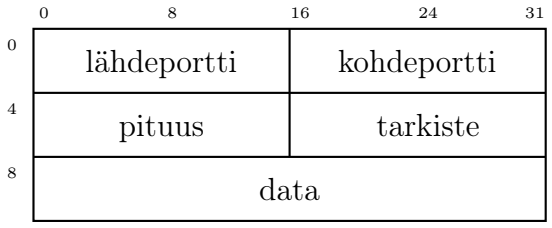
3. Jossakin vaiheessa myös toinen osapuoli on lähettänyt kaiken haluamansa datan, jolloin se lähettää dataa sisältämättömän segmentin, jossa FIN-lippu on aktiivinen.

4. Tässä vaiheessa molemmat osapuolet ovat ilmaisseet tahtovansa sulkea yhteyden. Alkuperäisen aloitteen tehnyt osapuoli kuittaa segmentin, ja yhteyden katsotaan sulkeutuneen. [5]

4.5 UDP-protokolla

User Datagram Protocol, eli UDP, on vaihtoehto luvussa 4.4 käsitellylle TCP:lle. TCP:n keskittyessä luotettavaan yhteyteen, UDP tarjoaa nopeampaa ja kevyempää tiedonsiirtoa. Se on yhteydetön protokolla, mikä tarkoittaa, ettei lähettäjä tiedä, toimitettiinko sähke vastaanottajalle asti. Yhteydettömyyden myötä monilähetykset ovat mahdollisia, toisin kuin TCP:ssä. UDP käyttää TCP:n tapaan 16-bittisiä portteja, joita käsitellään tarkemmin luvussa 4.6. [7]

UDP-sähkeen rakenne on esitetty kuvassa 17. Se on TCP-segmenttiin verrattuna hyvin yksinkertainen, koostuen ainoastaan viidestä kentästä: lähde- ja kohdeporteista, pituudesta, tarkisteesta ja datasta. Pituus määrittää koko sähkeen pituuden, sisältäen sekä otsikon että datan. Tarkiste on pakollinen kenttä, mutta sen käyttäminen ei ole pakollista. Datakenttä voi TCP:n tapaan olla tyhjä, jolloin kehyksen minimipituus on kahdeksan oktettia. [5, 7]



Kuva 17: UDP-sähkeen rakenne. [5]

4.6 TCP:n ja UDP:n käyttämät portit

Sekä TCP että UDP käyttävät portteja, joiden numeerinen arvo voi olla välillä 1...65535. Vaikka niiden lukuarvot ovat samalla alueella, niitä voidaan käyttää päällekkäin: samaa numeroa voidaan käyttää sekä UDP:ssa että TCP:ssä. Tämä johtuu siitä, että porttia sähkeen tai segmentin kohdeporttia tulkittaessa tiedetään jo, onko kyse UDP:sta vai TCP:stä. Portit ovat IP-osoitteiden tapaan IANA:n [14] hallinnoimia ja ne jakautuvat kolmeen ryhmään: tunnettuihin, rekisteröityihin ja vapaasti käytettäviin portteihin.

Portit 1 ... 1023 kuuluvat tunnettuihin portteihin. Niitä tulee käyttää ainoastaan IANA:n määrittämään tarkoitukseen. Esimerkiksi internetsivuja kuljettavalle HTTP-protokollalle on osoitettu portti 80. Tunnetun portin myötä käyttäjän ei tarvitse erikseen määrittää kohdeporttia, vaan selain tietää automaattisesti, mihin yhdistää. Rekisteröitävät portit kuuluvat välille 1024 ... 49151. Ne ovat tunnettujen porttien tapaan IANA:n määrittämiä ja niiden käyttäminen on luvanvaraista. Erona tunnettuihin portteihin on se, että mikäli jossakin verkossa ei käytetä lainkaan rekisteröityjä portteja, koko niille varattua aluetta voidaan käyttää mihin tarkoitukseen tahansa. Vapaasti käytettävien porttien alue muodostuu porteista 49152 ... 65535. IANA ei rajoita näiden käyttöä millään tavalla. [7]

4.7 Osoitteiden määrittäminen DHCP-palvelimen avulla

DHCP, eli *Dynamic Host Configuration Protocol*, on protokolla, jota käytetään IP-osoitteiden dynaamiseen määrittämiseen. Se vaatii toimiakseen DHCP-palvelimen, joka määrittää osoitteet keskitetysti. Sen etuna on, että verkkoon kytkevien laitteiden osoitteita ei tarvitse konfiguroida etukäteen, vaan konfigurointi tapahtuu automaattisesti. Samalla inhimillisen virheen mahdollisuus jää pois. Huonona puolena koko verkon toiminta riippuu DHCP-palvelimesta. Palvelin voidaan konfiguroida myös siten, että tietylle laitteelle osoitetaan aina sama IP-osoite. Asiakkaan kannalta tämä ei muuta toimintaperiaatetta. DHCP käyttää UDP-protokollaa, ja portteja 67 sekä 68. [7, 10]

IP-osoiteesta sovitaan neljässä vaiheessa. Ensimmäisessä vaiheessa DHCP:ta käyttävä laite kytketään verkkoon, jolloin sen on ensin etsittävä DHCP-palvelin. Laite ei tässä vaiheessa tiedä omaansa, eikä DHCP-palvelimen osoitetta, joten sen on etsittävä pal-

velinta monilähetysosoitteeseen 255.255.255.255 lähetettävällä sähkeellä. Lähettäjän osoitteena käytetään osoitetta 0.0.0.0. Sähkeen englanninkielinen nimi on *DHCP discover*, joka viittaa DHCP-palvelimen etsimiseen. Mikäli palvelinta ei tavoiteta, sähke lähetetään uudelleen ensin muutaman sekunnin välein. Mikäli palvelinta ei vieläkään tavoiteta, lähetysväliä harvennetaan viiteen minuuttiin. Etsimistä jatketaan niin pitkään, että DHCP-palvelin vastaa.

Toisessa vaiheessa DHCP-palvelin vastaa asiakkaan lähettämään kehykseen. Se varaa asiakkaalle jonkin IP-osoitteen ja ehdottaa sitä asiakkaalle sähkeellä. Palvelin tietää vastaanottamansa pyynnön perusteella asiakkaan MAC-osoitteen ja lähettää sähkeen suoraan sille. Tämän sähkeen englanninkielinen nimi *DHCP offer* viittaa tarjottuun IP-osoitteeseen. Kolmannessa vaiheessa asiakas vastaa palvelimen tarjoukseen. Mikäli DHCP-palvelimia on useita, asiakas tarttuu ensimmäiseen vastaanottamaansa tarjoukseen. Vastauksessa käyttäen yhä *DHCP discover* -tyyppistä monilähetystä, jotta mahdolliset muut palvelimet voivat vapauttaa varaamansa IP-osoitteen. Viimeisessä vaiheessa palvelin kuittaa asiakkaan pyynnön *DHCP ACK* -viestillä.

DHCP-palvelimen jakamilla osoitteilla voi olla voimassaoloaika. Tällöin asiakaslaite pyytää palvelinta uusimaan voimassaoloajan, kun siitä on kulunut puolet. Mikäli asiakkaan *DHCP REQUEST* -pyyntö tavoittaa palvelimen, se tyypillisesti hyväksytään. Jos palvelin kuitenkin vastaa kieltävästi *DHCP NACK* -viestillä, on prosessi aloitettava alusta. Laite, joka on menettänyt oikeuden käyttää aiemmin saamansa osoitetta, yrittää neuvotella itselleen saman osoitteen uudelleen. [7]

4.8 Osoitteiden dynaaminen määrittäminen ilman palvelinta

Microsoft on kehittänyt menetelmän tilanteisiin, jossa osoitteen määrittäminen DHCP-palvelimen kanssa ei onnistu. Se on nimeltään *Automatic Private IP Addressing*, ja lyhennetään APIPA. Nimi viittaa IP-osoitteiden automaattiseen määrittämiseen. APIPA ei tarkalleen ottaen ole TCP/IP:n mukainen protokolla, vaikka sitä tässä yhteydessä käsitelläänkin. Vaikka APIPA:a ei todennäköisesti tulla käyttämään lopputuotteissa, sitä voidaan hyödyntää tämän työn soveltavassa osuudessa.

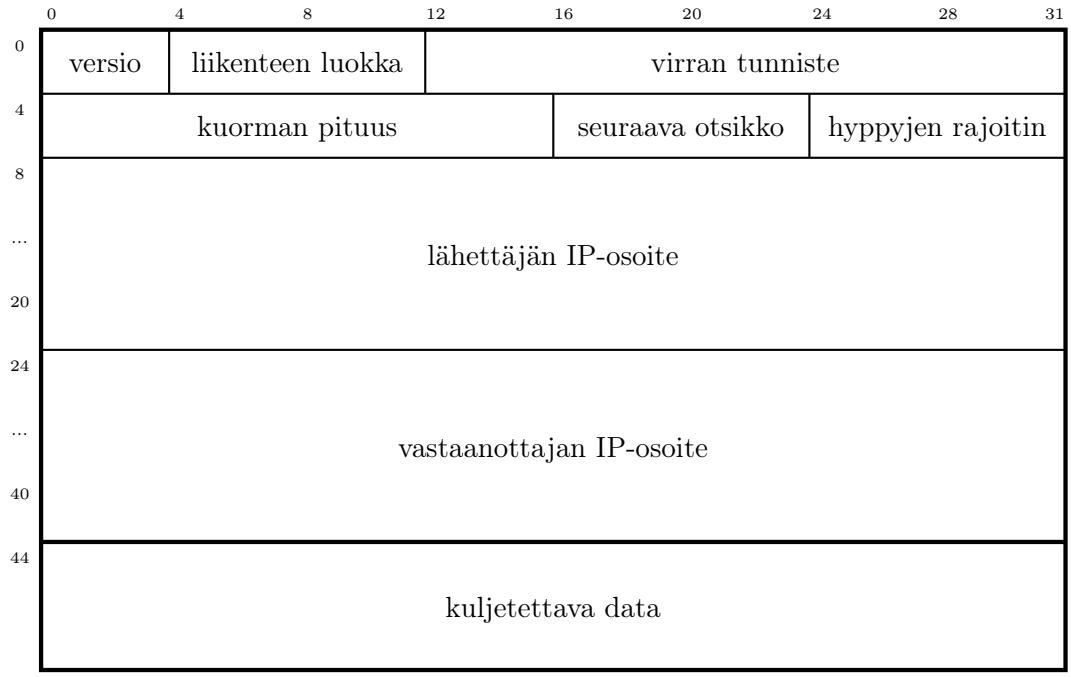
APIPA toimii siten, että jokainen menetelmää tukeva laite valitsee itselleen satunnaisen IP-osoitteen 168.254.0.0/16 -osoiteavaruudesta. Ennen osoitteen käyttöönottamista laite varmistaa ARP-protokollan avulla, ettei mikään toinen laite ole generoinut itselleen samaa osoitetta. Mikäli osoite on jo jollakin toisella laitteella, valitaan uusi satunnainen osoite. Menetelmän huono puoli on se, että kommunikointi on mahdollista ainoastaan muiden APIPA:a tukevien laitteiden kanssa. [15]

4.9 IPv6:n tuomat muutokset

Vaikka IPv4-protokolla on todistanut olevansa luotettava ja teknisesti toimiva, Internetin räjähdysmäisen kasvun myötä sen 32-bittinen osoiteavaruus on loppumassa kesken. IPv6:n tärkein uudistus onkin sen 128-bittinen osoiteavaruus. Sen

2^{128} yksilöllisen osoitteen myötä osoitteiden loppuminen on hyvin epätodennäköistä. Osoiteavaruuden koosta saa käsityksen, kun se suhteutetaan Maan pinta-alaan: mikäli osoitteet jaettaisiin tasaisesti Maan pinnalle, jokaiselle neliömetrille riittäisi yli 600 000 000 000 000 000 000 000 osoitetta. Osoitteet esitetään kuutena neljän heksadesimaaliluvun ryhmänä, jotka erotetaan toisistaan kaksoispisteellä. Mikäli ryhmän korkeimmat tavut ovat arvoltaan nollia, ne voidaan jättää merkitsemättä. Lisäksi pelkkiä nollia sisältävät ryhmät voidaan jättää kokonaan merkitsemättä, jolloin ne korvataan kahdella peräkkäisellä kaksoispisteellä. Niinpä osoitteet 1080:0000:0000:0000:0008:0800:200C:417A, 1080:0:0:0:8:800:200C:417A ja 1080::8:800:200C:417A tarkoittavat kaikki samaa. IPv6:sta käytetään toisinaan epävirallista nimeä IPng. Nimi muodostuu englanninkielisistä sanoista *IP Next Generation*, tarkoittaen seuraavan sukupolven IP:tä. [6, 15]

IPv6 uudistaa osoitteiden lisäksi myös paketin rakenteen. Uudistettu rakenne on esitetty kuvassa 18 ja se koostuu versiosta, liikenteen luokasta, virran tunnisteesta, kuorman pituudesta, seuraavasta otsikosta, hyppyjen rajoittimesta, lähettäjän ja vastaanottajan IP-osoitteista sekä kuljetettavasta datasta. Paketin alussa on IPv4:n tapaan neljän bitin kenttä, joka määrittää käytettävän version. IPv6:n tapauksessa se saa aina arvon 6. Tämän jälkeen alkavat eroavaisuudet. IPv6 tukee pakettien priorisointia liikenteen luokka -kentän avulla. Priorisoinnin avulla reititin voi myös valikoida hylättävät paketit verkon ylikuormitustilanteissa. Käytännössä priorisointia ei juuri käytetä, jolloin kenttä on numeeriselta arvoltaan nolla. Reitittimien kuormitusta pienennetään virran tunnisteella. Se nopeuttaa suurten, useampiin fragmentteihin pilkottavien, datamäärien käsittelyä. Kun jokainen fragmentti merkitään samalla



Kuva 18: IPv6-protokollan mukainen paketti. Vaakasuunnan numerot ovat bittien, ja pystysuunnan numerot oktettien järjestysnumeroita. [5]

Taulukko 10: Luettelo IPv6:n yleisimpiä protokollia vastaavista numeroista. Luvut on esitetty sekä 10-kantaisina että 16-kantaisina. [5]

Protokolla	10-kantainen arvo	16-kantainen arvo
TCP	6	0x06
UDP	17	0x11
ICMP	58	0x3A
Ei seuraavaa otsaketta	60	0x3C

virran tunnisteella, reititin voi hyödyntää ensimmäisenä lähetetyn fragmentin tietoja käsitellessään muita samaan virtaan kuuluvia fragmentteja. Yksittäisen paketin koko määritellään kuorman pituus -kentässä. Se on IPv4:n tapaan kahden tavun kokoinen ja käyttää yksikkönään tavua, mutta poikkeaa siten, että IPv6:ssa otsakkeen kenttiä ei lasketa pituuteen. Seuraava otsikko -kenttä muistuttaa IPv4:n protokolla-kenttää: se määrittelee pakettiin kapseloidun protokollan. IPv6 käyttää kenttää kuitenkin monipuolisemmin, sillä paketin otsaketta voidaan laajentaa seuraava otsikko -kentän avulla. Esimerkiksi paketin pirstoutuessa useampaan fragmenttiin otsakkeeseen liitetään fragmentin kehys. Jokaisella pakettia laajentavalla otsakkeella on oma, vastaava kenttä, joka määrittelee seuraavan otsikon tyypin. Niinpä laajennuksia voidaan ketjuttaa. Taulukossa 10 on esitetty joitakin yleisimpiä IPv6:n protokollia vastaavat arvot. [5, 15]

IPv6:ssa ei ole erillistä ARP-protokollaa, vaan osoitteiden selvittäminen, verkon diagnosointi sekä virhetilainteiden käsittely ovat IPv6:ssa ICMP-protokollan tehtäviä. DHCP-protokollaa on uudistettu tukemaan IPv6:n pidempiä osoitteita. Muutosten myötä edellä mainittujen protokollien kehykset ovat muuttuneet, mutta niiden toimintaperiaate ja tarkoitus ovat ennallaan. [5]

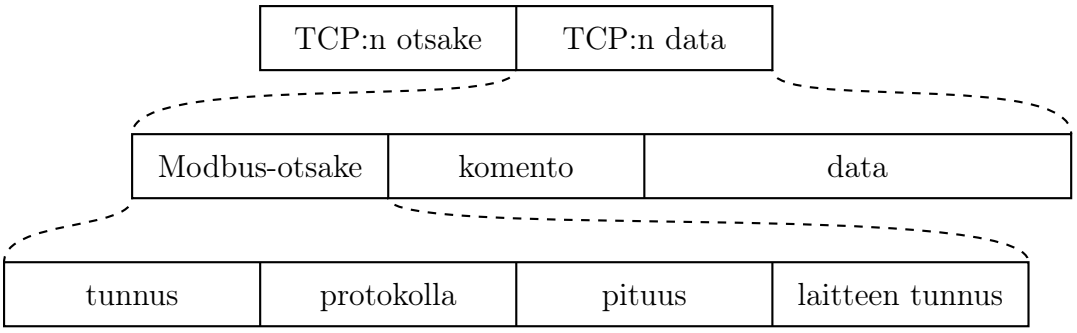
5 Ethernet teollisuudessa

Ethernet kehitettiin alun perin toimistokäyttöön, eikä teollisuuden tarpeita siksi otettu huomioon. Teollisuuden vaatimukset poikkeavat toimiston tarpeista merkittävästi esimerkiksi olosuhteiden, luotettavuuden ja vasteaikojen kannalta. Ethernetin yleistymisen myötä myös teollisuuden tarpeisiin soveltuvia ratkaisuja on kehitetty. Hankalampien olosuhteiden myötä käyttöön on otettu mm. suojattuja kaapeleita ja liittimiä, jotka kestävät mekaanista rasitusta ja suojaavat ulkoisilta uhilta, kuten pölyltä ja nesteiltä. Tavallisessa Ethernet-verkossa kapasiteetista voidaan hyödyntää 25 % ... 30 %, kun taas teollisessa verkossa kuormitusaste on vasteaikojen takaamiseksi tyypillisesti alle 10 % nimellisestä kapasiteetista. Vaikka verkon kokonaiskapasiteetti riippuu sen nopeudesta, teollisuudessa ei yleensä käytetä 1000 Mb/s siirtonopeuteen kykeneviä 1000Base-T -verkkoja. Teoreettiselta suorituskyvyltään kymmenen kertaa hitaampi 100Base-TX on tyypillisempi ratkaisu kaksisuuntaisuutensa vuoksi. Tällöin verkossa ei voi tapahtua törmäyksiä, ja sen vasteajat voivat olla jopa lyhempiä. Lisäksi teollisissa verkoissa käytetään tähden lisäksi myös muita topologioita, esimerkiksi rengasta.

Vaikka luvussa 4 käsitelty TCP/IP -protokollaperhe tarjoaa yhdessä Ethernetin kanssa avoimen pohjan aina sovelluskerrokselle asti, sen sovelluskerros ei aina sovellu teolliseen käyttöön. Siksi monet valmistajat käyttävät omia sovellustason ratkaisujaan, mutta myös avoimia ratkaisuja on olemassa. Osa ratkaisuista käyttää TCP/IP:tä ja standardia Ethernetiä, mutta jotkut rikkovat yhteensopivuuden paremman suorituskyvyn toivossa. Tässä luvussa luodaan katsaus joihinkin teollisiin lähiverkkoratkaisuihin. Tekniset yksityiskohdat jätetään pienelle huomiolle, sillä tarkoituksena on luoda yleiskatsaus teollisiin ratkaisuihin. [3, 11]

5.1 Modbus/TCP

Modbus oli alun perin sarjaliikenneprotokolla, jota on käytetty niin radioliikenteessä kuin perinteisissä sarjaporteissakin. Se on joidenkin arvioiden mukaan yleisin automaatiojärjestelmissä käytetty protokolla. Modbusin suosioon on vaikuttanut sen avoimuus ja ilmaisuus: protokollan spesifikaatio on saatavilla Modbusin verkkosivuilta [16] maksutta, eikä Modbusin käytöstä tarvitse maksaa rojalteja. Ethernetin yleistyttyä Modbusista kehitettiin Modbus/TCP, joka käyttää siirtotienään Ethernetiä. Modbus/TCP käyttää pääpiirteittäin samaa rakennetta kuin alkuperäinen Modbus. Sen kehyksiä siirretään nimensä mukaisesti TCP-segmenteissä. Koska TCP on jo itsessään luotettava protokolla, tarkisteesta vastaava kenttä on jätetty Modbus/TCP:stä pois. Modbus/TCP:n kehys koostuu kuvan 19 mukaisesti kolmesta kentästä: otsakkeesta, komennosta ja datasta. Lisäksi otsake määrittelee tapahtuman tunnuksen, käytettävän protokollan, kehyksen pituuden ja laitteen tunnuksen. Näistä protokolla ja laitteen tunnus ovat tarpeen silloin, kun käytetään Ethernetin ja perinteisen sarjaliikenneväylän välistä siltaa. Tällöin saman IP-osoitteen takana voi olla useita laitteita. [11, 17]



Kuva 19: Modbus/TCP-protokollan kehys kapseloidaan tavalliseen TCP-segmenttiin. [17]

Modbusin kehykset jakautuvat kolmeen eri luokkaan: pyyntöihin (*request*), vastauksiin (*responce)* ja poikkeuksiin (*exception responce)*. Se mallintaa dataa neljällä erityyppisellä objektilla. Yksittäinen bitti, jonka tila voidaan lukea väylän kautta, on nimeltään *discrete input*, eli yksittäinen tulo. Mikäli bitin tila voidaan sekä lukea että kirjoittaa väylän kautta, sen nimitys on *coil*, eli kela. Vastaavat nimitykset 16-bittisille objekteille ovat *input register* (tulorekisteri) ja *holding register* (pitorekisteri). Kommunikointi perustuu näiden objektien lukemiseen ja kirjoittamiseen. Komentojen 8-bittinen alue on jaettu standardoituihin ja toteutuskohtaisiin komentoihin. [18]

Modbus/TCP:n hyviä puolia ovat alkuperäisen Modbusin yleisyys, yhteensopivuus Ethernetin ja TCP/IP-protokollaperheen kanssa sekä sen avoimuus ja ilmaisuus. Se on kuitenkin kehitetty alun perin sarjamuotoista liikennettä varten, ja olettaa, että väylällä on aina yksi isäntälaite. Monissa tapauksissa Modbus/TCP:n ominaisuudet ovat riittäviä, mutta historiansa vuoksi se ei hyödynnä koko Ethernetin tarjoamaa potentiaalia. Useampia pienikokoisia rakenteita voidaan toki pakata yhteen 16 bitin objektiin ja vastaavasti suuria rakenteita pilkkoa useampiin eri objekteihin, mutta kahteen objektikokoon rajoittuminen on Ethernetin kapasiteetin valossa tarpeetonta. Useampia objektityyppejä ei ole luotu, jotta yhteensopivuus alkuperäiseen Modbus-protokollaan säilyy. [11]

5.2 EtherNet/IP

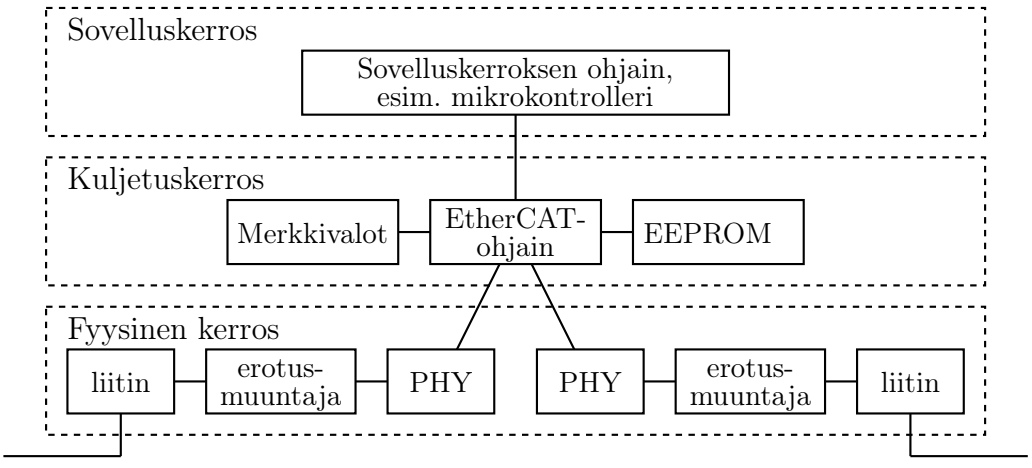
EtherNet/IP käyttää sovelluskerroksenaan Open DeviceNet Vendor Associationin, eli ODVA:n [19], ylläpitämää CIP-protokollaa. Lyhenne muodostuu englanninkielisistä sanoista *Common Industrial Protocol* ja tarkoittaa yleistä teollista protokollaa. Samaan perheeseen kuuluu EtherNet/IP:n lisäksi DeviceNet, CompoNet ja ControlNet. Kaikki neljä käyttävät samaa CIP-protokollaa sovelluskerroksellaan, mutta alempien kerrosten ratkaisut poikkeavat toisistaan. EtherNet/IP käyttää alemmilla kerroksilla TCP/IP-protokollaa ja IEEE 802.3 mukaista Ethernetiä. Lisäksi fyysiselle kerrokselle on olemassa suosituksia, joita ei ole kuitenkaan pakko noudattaa. EtherNet/IP:tä käyttäville laitteille on tehtävä pakollinen yhteensopivuustesti, jolla varmistetaan, että laite toteuttaa EtherNet/IP:n standardien vaatimalla tavalla. [3, 20].

Myös CIP:n kommunikointi perustuu objekteihin. Ne voidaan jakaa kolmeen eri tyyppiin, jotka ovat pakolliset objektit (*Required Objects*), sovelluksen objektit (*Application Objects*) ja valmistajakohtaiset objektit (*Vendor-specific Objects*). Sulkeissa on käytetty tyyppien virallisia, englanninkielisiä nimiä. Pakolliset objektit on toteutettava kaikkiin CIP:iä käyttäviin laitteisiin. Laitteen tunnus on yksi esimerkki pakollisesta objektista. Lisäksi CIP tukee luokkia, jotka rakentuvat useista objektista. CIP käyttää kommunikointiin kahta erilaista viestityyppiä: eksplisiittisiä ja implisiittisiä viestejä. Eksplisiittiset viestit soveltuvat pyyntöihin ja vastauksiin, jotka eivät ole aikakriittisiä. Laitteen parametrien muuttaminen on tyypillinen esimerkki eksplisiittisestä viestistä: parametrien muuttaminen ei ole aikakriittistä, mutta on tärkeää, että parametrit välittyvät perille luotettavasti. Implisiittiset viestit taas soveltuvat reaaliaikaisen datan siirtämiseen. EtherNet/IP:n tapauksessa eksplisiittiset viestit käyttävät TCP-protokollaa ja implisiittiset UDP-protokollaa. Dataa voidaan joko pyytää kiertokyselyn avulla (*polled data*), sitä voidaan lähettää säännöllisesti (*cyclic data*) tai tietyn tapahtuman yhteydessä (*event-driven data*).

Jokaisesta laitteesta luodaan EDS-tiedosto, jossa kuvataan laitteen ominaisuudet. Tätä tiedostoa käytetään laitteen ja koko verkon konfigurointiin. Se voidaan myös sisällyttää laitteeseen yhtenä objektina. Hyvin tehdyn EDS-tiedoston avulla laitteen käyttöönotto ja järjestelmään liittäminen helpottuvat merkittävästi. [11, 20].

5.3 EtherCAT

EtherCAT on isäntä-orja -tyyppinen kenttäväylä, joka käyttää siirtotienään Ethernetiä. Isäntälaite ei poikkea laitteiston osalta tavallisesta Ethernetistä, mutta orjalaitteiden toimintaperiaate poikkeaa siitä hieman: niissä käytetään vähintään kahta Ethernet-porttia, joita ohjaa kuvan 20 mukaisesti erityinen EtherCAT-ohjainpiiri.



Kuva 20: EtherCAT-orjan lohkokaavio. EtherCAT käyttää siirtotienään standardinmukaista Ethernetiä ja on yhteensopiva TCP/IP:n kanssa. Orjalaitteissa on tyypillisesti kaksi Ethernet-porttia, jolloin verkko tukee useita eri topologioita. [21]

Tämän myötä EtherCAT tukee tavalliselle Ethernetille tyypillisen tähden lisäksi myös muita verkon topologioita. Tyypillisesti portteja on kaksi, ja verkko on topologialtaan väylä. Parikaapeliverkoissa EtherCAT käyttää 100Base-TX -standardia, jolloin sen kapasiteetti on 100 Mb/s sekä lähettämiselle että vastaanottamiselle.

EtherCAT siirtää dataa tyypillisesti Ethernet II -tyypin kehyksissä, jolloin sen tyyppikenttä saa arvokseen 0x88A4. Oman tyyppinumeron myötä samassa verkossa voidaan kuljettaa myös muuta dataa, esimerkiksi TCP/IP:n mukaisia paketteja. Samassa kehyksessä voi olla useille eri orjalaitteelle kuuluvaa dataa. Kukin laite tunnistaa itselleen kuuluvan osion ja lähettää kehyksen muilta osin muuttumattomana eteenpäin. Kehyksiä käsitellään ja lähetetään orjalaitteissa reaaliaikaisesti, jo ennen kuin kehys on kokonaan saapunut. Vaikka prosessointi aloitetaan vaillinaisella kehyksellä, laitteet eivät voi suorittaa mitään kehyksen sisältämää toimintoa ennen kuin kehys on kokonaisuudessaan vastaanotettu, sillä kehyksen tarkiste on vasta kehyksen lopussa. Reaaliaikainen kehyksen prosessointi ja synkronoidut kellosignaalit tekevät EtherCATin vasteajoista helposti ennustettavia.

Jokaisesta orjalaitteesta luodaan ESI-tiedosto, joka kuvaa EtherCAT-orjan ominaisuuksia. Näiden avulla luodaan koko verkkoa kuvaava ENI-tiedosto, jonka avulla verkon isäntälaite kommunikoi koko verkon kanssa. Tämä käytäntö muistuttaa Ethernet/IP:n EDS-tiedostoja. Yhteneväisyyksiä on enemmänkin, sillä myös EtherCAT-laitteille on tehtävä pakollinen yhteensopivuustestaus. Koska EtherCAT on suunniteltu nimenomaan teollisuuden tarpeisiin, se soveltuu ominaisuuksiensa puolesta teollisuuteen tavanomaista Ethernetiä paremmin. Uudet ominaisuudet kuitenkin vaativat erityisen ohjainpiirin käyttämistä orjalaitteissa, mitä on pidettävä huonona puolena yhteensopivuutta ajatellen. [21, 22]

5.4 PROFINET

PROFINET on avoin, teollisen Ethernetin standardi, joka määritellään standardeissa IEC 51158 ja IEC 61784. Se käyttää siirtotienään 100 Mb/s kaksisuuntaista Ethernetiä. Myös valokuidun ja tietyissä rajoissa langattomienkin siirtoteiden käyttäminen on mahdollista. Verkkoon kytkettävien laitteiden osalta PROFINET on yhteensopiva perinteisen Ethernetin kanssa, mutta verkon rakentamiseen käytettävät kytkimet riippuvat käytettävästä PROFINET:n versiosta. PROFINET mahdollistaa reaaliaikaisen tiedonsiirron ja parantaa luotettavuutta eri verkon topologioiden avulla. Yksi PROFINET:n tärkeimmistä tavoitteista on tehdä verkon käyttöönotosta ja ylläpidosta helppoa, ja sitä kautta edullista. PROFINET:n kehityksestä ja ylläpidosta vastaa PROFIBUS & PROFINET International. PROFIBUS on saman organisaation aiemmin kehittämä sarjaliikenneprotokolla. Organisaatiosta käytetään yleisesti myös lyhennettä PI.

Verkkoon voi kuulua kolmenlaisia laitteita. Näiden englanninkieliset nimet ovat IO controller, IO device ja IO supervisor. Näistä IO controller ja IO device muodostavat toiminnallisen osuuden, ja IO supervisoria käytetään ainoastaan käyttöönottoon ja

verkon diagnosointiin. IO device -tyyppiset laitteet tarjoavat järjestelmälle toimintoja, joita vastaa englanninkielinen termi *slot*. IO controller -tyyppiset laitteet ohjaavat edellä mainittuja toimintoja. Laitetyyppien lisäksi PROFINET-laitteet jaetaan kolmeen eri luokkaan. Luokkien nimet ovat englanninkielisten sanojen *Conformance Class* mukaisesti CC-A, CC-B ja CC-C. CC-A -luokan laitteet toteuttavat vain perustoiminnot. Tämä on myös ainoa luokka, jossa voidaan käyttää langatonta tiedonsiirtoa. CC-B:ssä verkosta saadaan diagnostiikkatietoja, ja laitteet tuntevat verkon topologian. CC-C:ssä osa verkon kapasiteetista varataan synkronoidulle kommunikoinnille. Tämä toteutetaan jakamalla verkon kapasiteetti kahteen vuorottelevaan aikaikkunaan. Näistä toinen palvelee synkronoitua, reaaliaikaista tiedonsiirtoa, ja toinen asynkronista TCP/IP:tä.

Koska PROFINET käyttää tavallista Ethernetiä, se käyttää sekä MAC- että IP-osoitteita. Tämän lisäksi jokaiselle PROFINET-laitteelle osoitetaan yksilöllinen nimi. Se joko osoitetaan laitteelle käyttöönoton yhteydessä, tai laite voi päätellä sen automaattisesti tunnistamalla verkosta itseään lähellä olevat laitteet. Automaattinen tunnistaminen edellyttää, että laite tuntee verkon topologian. Menetelmä tunnetaan sen englanninkielisellä nimellä *neighborhood detection*, ja sitä käytetään myös, mikäli laite vaihdetaan jostakin syystä uuteen. Nimeä käytetään IP-osoitteen määrittämiseen, mutta myös muiden menetelmien, kuten DHCP:n, käyttäminen on mahdollista. Mikäli valmistaja ei tahdo hankkia omaa OUI-lohkoa IEEE:ltä, PROFIBUS & PROFINET International tarjoaa 4000 osoitteen lohkoja PROFINET-laitteiden käyttöön. Laitteiden yhteensovipuuden varmistamiseksi PROFINET-laitteiden on läpäistävä pakollinen yhteensopivuustestaus. [23]

6 Ohjainlaitteen suunnitteleminen

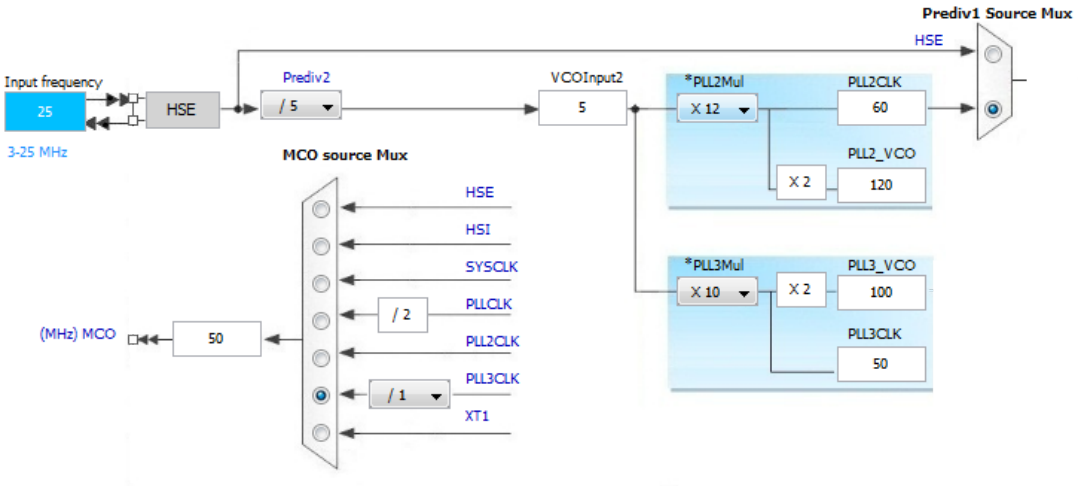
Soveltavan osuuden tavoitteena on kehittää Ethernetin kautta hallittava sulautettu järjestelmä, jolla voidaan ohjata viittä eri kanavaa. Kuhunkin kanavaan kytketään erivärinen LED. Ohjattavat värit ovat punainen, vihreä, sininen, sekä kaksi erisävyistä valkoista. Järjestelmän tulee toimia ilman reaaliaikaista käyttöjärjestelmää ja sen hallintaan käytetään erillistä sovellusta. Käyttöliittymän kehittämiseen päätettiin käyttää Qt:ta sen alustariippumattomuuden vuoksi. Vaikka tämä työ keskittyykin parikaapelipohjaiseen Ethernetiin, sama sovellus voidaan helposti muokata toimimaan esimerkiksi älypuhelimella. Sovelluksen tekninen toteutus on rajattu työn ulkopuolelle, ja sitä käsitellään vain käyttöliittymän ja tuotetun verkkoliikenteen kautta. Koska aiempaa kokemusta ARM-prosessoreiden kehitysympäristöistä ei ole, soveltuva kehitysympäristö on ensin etsittävä. Teknowaren lopputuotteet ovat asiakaskohtaisesti räätälöityjä, joten laitteen alustaksi valittiin julkisesti saatavilla oleva Olimex STM32-P107 -kehitysalusta [24].

Jotta sovelluksen käyttäminen olisi mielekästä, sen täytyy löytää ohjainlaite automaattisesti kaikissa olosuhteissa. Tämän seurauksena molempien osapuolten on tuettava myös APIPA-menetelmää, sillä DHCP-palvelinta ei välttämättä ole aina saatavilla. Lisäksi käyttöliittymään on lisättävä myös mahdollisuus yrittää kohdeosoitteen määrittelemistä käsin, mikäli automaattinen yhdistäminen epäonnistuu. Mikäli yhteys ohjattavaan laitteeseen katkeaa, ohjelman tulee palata aloitusnäkymään ja yrittää muodostaa yhteys uudelleen. Yhdistämisen jälkeen sovelluksen tulee kommunikoida TCP-yhteyden kautta ohjainlaitteen kanssa. Koska mitään erillistä sovelluskerroksen pinoa ei tahdota vielä tässä vaiheessa hankkia, TCP:n käyttämä dataformaatti kehitetään itse. Kunkin kanavan kirkkauden on oltava helposti ja tarkasti säädettävissä hallintaohjelmiston käyttöliittymästä.

6.1 STM32-P107 kehitysalusta

Työn soveltava osuus perustuu Olimexin STM-P107 -kehitysalustaan, joka pohjautuu ST Microelectronicsin STM32-F107 -mikrokontrolleriin. Käytössä oli alustan C-versio. Se koostuu useista lohkoista, joista tämän työn kannalta olennaisimpia ovat Ethernet-liitäntä kommunikointia ja JTAG-liitäntä mikrokontrollerin ohjelmointia varten. Lisäksi alustalta hyödynnettiin RS232-väylää ja merkkivaloja laitteen tilan seuraamiseen. Alustaa voi käyttää joko 6,5 V ... 9,0 V tasajännitteellä, tai 6 V vaihtojännitteellä. Tässä työssä sitä käytettiin tasajännitteellä. Mikrokontrollerin ohjelmointiin käytettiin ST Microelectronicsin ST-Link/V2 -ohjelmointilaitetta [25].

Kehitysalustan mikrokontrolleri sisältää MAC-moduulin, joten erillistä MAC-piiriä ei tarvita. Alustan PHY-piirinä on Microchipin LAN8710A ja se on kytketty mikrokontrolleriin RMII-väylällä. Erotusmuuntaja ja Ethernetille tyypilliset merkkivalot on yhdistetty RJ45-liittimeen. Mikrokontrollerille on kytketty 25 MHz kide. Sen tuottama signaali muunnetaan mikrokontrollerin sisäisillä moduuleilla RMII:n vaa-



Kuva 21: Kuvakaappaus STM32CubeMX-ohjelmasta. RMII:n vaatima 50 MHz signaali muodostetaan 25 MHz kiteestä mikrokontrollerin MCO-pinniin sisäisten skaalaimien avulla.

timaksi 50 MHz signaaliksi kuvan 21 mukaisesti. Muunnos ei ole automaattisesti käytössä, vaan se täytyy valita ohjelmallisesti. [26]

6.2 Työssä käytetyt ohjelmistot

Sulautetun järjestelmän ohjelmistokehitykseen kokeiltiin useita eri kehitysympäristöjä. Kaupallisista kehitysympäristöistä IAR Embedded Workbench [27] ja Keil MDK [28] jouduttiin hylkäämään käytössä olleiden ilmaisversioiden rajoitteiden vuoksi. Lyhyeksi jääneen testaamisen perusteella ne osoittautuivat sinänsä toimiviksi, mutta käyttöliittymiltään hankaliksi ja hieman vanhahtaviksi kehitysympäristöiksi. Seuraavaksi käytössä oli AC6:n System Workbench for STM32 [29], josta käytetään myös lyhennettä SW4STM32. Se perustuu useisiin avoimen lähdekoodin projekteihin, kuten Eclipseen, GCC:hen ja GDB:hen. Käytössä oli kehitysympäristön versio 1.5.1. Järjestelmää kehitettiin pitkään SW4STM32:n avulla, ja kehitysympäristö toimikin pääosin moitteettomasti. Sen heikkoudeksi osoittautuivat yhteensopivuusongelmat STM32CubeMX-työkalun kanssa. Projektin luominen toimi odotetulla tavalla, mutta sen päivittäminen sai projektitiedostot korruptoitumaan siten, että projekti oli käytännössä luotava muutosten yhteydessä uudelleen. Myöhempää kaupallista käyttöä ajatellen on myös todettava, että tälle ympäristölle on saatavana ainoastaan yhteisöpohjaista tukea. Havaittujen ongelmien vuoksi kehitysympäristöä päätettiin vaihtaa, ja SW4STM32:n tilalle valittiin lopulta Atollic TrueSTUDIO [30]. Se perustuu SW4STM32:n tapaan Eclipseen ja GCC:hen, joten siirtyminen oli kohtuullisen yksinkertaista. Käytössä ollut TrueSTUDIOn ilmaisversio sisältää kaikki välttämättömät työkalut, eikä se rajoita ohjelman kokoa. Sen kaupallinen versio sisältää lisäksi lähdekoodin analysointiin ja ohjelman testaamiseen tarkoitettuja työkaluja, teknisen tuen sekä mahdollisuuden valita käytettävien työkalujen versio. Ilmaisversiossa

käytetään aina uusinta asennettua versiota. Teollisessa käytössä on tuotteiden ylläpidon kannalta tärkeätä, että myös työkalujen vanhemmat versiot ovat tarvittaessa käytettävissä.

Kaikkien kehitysympäristöiden projektit luotiin ST Microelectronicsin tarjoamalla STM32CubeMX-työkalun [31] versiolla 4.11. Se on graafinen työkalu, jolla voidaan generoida mikrokontrollerin alustavaa lähdekoodia ja luoda projekti varsinaista kehitysympäristöä varten. Lisäksi työkaluun on sisällytetty joitakin ohjelmistopinoja, kuten tässä työssä käytettävä lwIP-pino. Kehitysalustan mikrokontrolleri voidaan ohjelmoida suoraan kehitysympäristön käyttöliittymästä, mutta se vaatii toimiakseen tuen myös kontrollerille ladatusta ohjelmasta. Ongelmatilanteissa mikrokontrolleri nollattiin ohjelmointilaitteen oman *STM32 ST-LINK utility* -ohjelman [32] avulla. Verkon toisena osapuolena käytettiin tietokonetta, jonka käyttöjärjestelmä on Windows 7. Työn alkuvaiheessa verkon toimintaa testattiin Packet Sender -työkalulla [33]. Myöhemmin kommunikointiin kehitettiin Qt-pohjainen hallintaohjelma. Hallintaohjelman kehittäminen on rajattu työn ulkopuolelle, ja sitä käsitellään työssä vain sen tuottaman verkkoliikenteen ja käyttöliittymän kautta. Verkkoliikennettä tarkkailtiin Wireshark-ohjelman [34] versiolla 1.12.8.

6.3 LwIP-pino

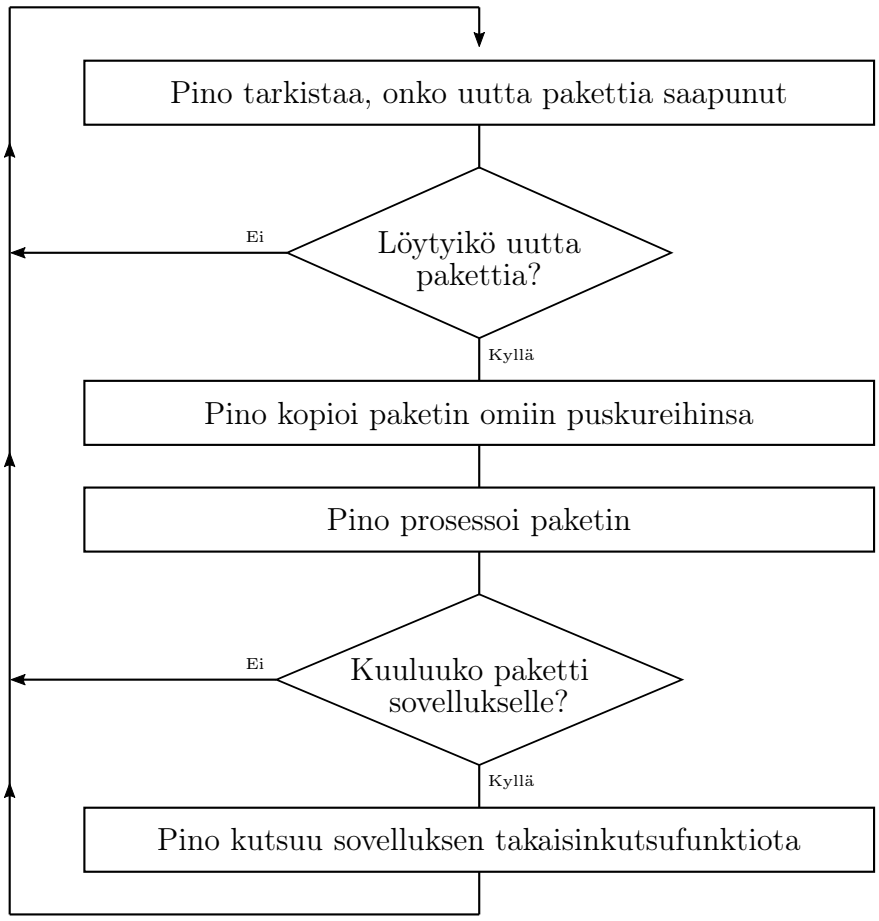
LwIP on Adam Dunkelsin kehittämä TCP/IP -pino, jonka ST Microelectronics on integroinut osaksi STM32CubeMX-työkaluaan. LwIP tukee verkkokerroksella mm. IPv4-, IPv6-, ICMP- ja IGMP-protokollia ja kuljetuskerroksella sekä TCP:tä että UDP:ta. Sovelluskerrokselta tuettuja ovat vain TCP/IP-perheen keskeisimmät protokollat, esimerkiksi DNS ja DHCP. LwIP:n käyttöön on olemassa kolme rajapintaa, mutta ilman reaaliaikaista käyttöjärjestelmää ainoastaan kolmikon yksinkertaisin rajapinta, *raw api*, on käytettävissä. Sen virallinen dokumentaatio [35] toimitetaan pinon mukana, mutta käytännössä esimerkiksi tämän kappaleen lähteenä käytettävät, ST Microelectronicsin tuottamat dokumentit, ovat selkeämmin jäsenneltyjä ja havainnollisempia, erityisesti STM32CubeMX-ohjelmaa käytettäessä. Pinon asetukset määritetään kahdessa eri otsikkotiedostossa, jotka ovat nimiltään *opt.h* ja *lwipopts.h*. Näistä ensin mainittu sisältää pinon oletusasetukset, joita voidaan muokata sovelluksen tarpeiden mukaan jälkimmäisen avulla. Suurin osa asetuksista voidaan määrittää STM32CubeMX-työkalun avulla graafisesta käyttöliittymästä.

LwIP:n virallinen julkaisu ei sisällä ajureita millekään rautatason moduuleille. Sen mukana toimitetaan *ethernetif.c* -niminen tiedosto, jota voidaan käyttää ajureiden pohjana. Sen tärkein funktio on nimeltään *ethernetif_input*, joka siirtää paketteja rautatason moduuleilta pinolle. Kun projekti luodaan STM32CubeMX-ohjelmalla, kyseinen tiedosto generoidaan automaattisesti ja sisältää ST Microelectronicsin kirjoittaman ajurin. [36, 37]

6.3.1 Kuljetuskerroksen rajapinnat

Raw api tarjoaa kuljetuskerrokselle sekä UDP:ta että TCP:tä tukevat rajapinnat. Saapuvaa dataa käsitellään käytettävästä protokollasta riippumatta kuvassa 22 esitetyllä tavalla. Pino tarkkailee säännöllisesti, onko uusia paketteja saapunut. Kun uusi paketti havaitaan, pino kopioi sen omiin puskureihinsa ja prosessoi sen. Mikäli paketti kuuluu jollekin pinon ulkopuoliselle sovellukselle, kutsutaan sille määrättyä takaisinkutsufunktiota. Kunkin sovelluksen takaisinkutsufunktio määritetään samalla, kun sovellus rekisteröidään pinolle. TCP:tä ja UDP:tä käyttäville sovelluksille on erilliset rajapinnat, jotka osittain muistuttavat toisiaan. Molempia hallitaan PCB:iksi kutsuttavien tietorakenteiden kautta. Lyhenne muodostuu englanninkielisistä sanoista *protocol control block*. Kullekin sovellukselle määritetään yksi tai useampia PCB:itä, joiden kautta se kommunikoi lwIP:n kanssa. [37]

UDP-protokolla on sovelluksen kannalta yksinkertaisempi toteuttaa. Sen rajapinnan muodostavat funktiot on esitelty taulukossa 11. Ohjelman alussa sovellukselle on luotava uusi PCB *udp_new* -funktiolla. PCB ei silti ole käytettävissä, ennen kuin se on kytketty joko paikalliseen IP-osoitteeseen ja porttiin *udp_bind* -funktiolla,



Kuva 22: Saapuvan paketin prosessointikaavio, kun käytössä on *raw api* -rajapinta. [37]

Taulukko 11: LwIP:n tarjoama rajapinta UDP-protokollalle. [37]

Funktio	Kuvaus
udp_new	Luo uuden UDP-PCB:n
udp_remove	Tuhoaa UDP-PCB:n
udp_bind	Kytkee UDP-PCB:n paikalliseen IP-osoitteeseen ja porttiin
udp_connect	Määrittää kohteen IP-osoitteen ja portin
udp_disconnect	Poistaa kohteen IP-osoitteen ja portin UDP-PCB:ltä
udp_send	Lähettää UDP-sähkeen
udp_recv	Määrittää sovelluksen käyttämän takaisinkutsufunktion datan vastaanottamista varten

tai kohteen IP-osoitteeseen ja porttiin *udp_connect* -funktiolla. Datan vastaanottamista varten on käytettävä *udp_bind* -funktiota ja määritettävä takaisinkutsu *udp_recv* -funktion avulla. Lähettämistä varten taas on käytettävä *udp_connect* -funktiota, minkä jälkeen UDP-sähkeitä voidaan lähettää *udp_send* -funktion avulla. Mikäli PCB:n käyttäminen halutaan lopettaa, sen kohdeosoite ja -portti voidaan poistaa *udp_disconnect* -funktiolla ja koko PCB:n käyttämä muistiavaruus voidaan vapauttaa *udp_remove* -funktiolla. Kuten luvussa 4.5 todettiin, UDP on yhteydetön protokolla, eikä siten vaadi yhteyden avaamista tai sulkemista. Niinpä *udp_connect* ja *udp_disconnect* -funktiot eivät nimestään huolimatta tuota lainkaan verkkoliikennettä, vaan vaikuttavat ainoastaan pinon sisäiseen toimintaan. [35, 37]

TCP-protokollan rajapinta on hieman monimutkaisempi kuin UDP-protokollan. Sitä voidaan käyttää kahdella eri tavalla: joko asiakkaana, tai palvelimena. TCP-protokollan rajapinta on koottu taulukkoon 12. Molemmissa tapauksissa sovelluksen on luotava uusi PCB *tcp_new* -funktiolla. Mikäli sovellus toimii asiakkaana, yhteys muodostetaan *tcp_connect* -funktiolla. Palvelimena toimivien sovellusten taas tulee käyttää ensin *tcp_bind* -funktiota paikallisen osoitteen ja portin määrittämiseen, minkä jälkeen on määritettävä takaisinkutsu saapuvia yhteyksiä varten *tcp_accept* -funktiolla. Viimeisenä vaiheena on kutsuttava *tcp_listen* -funktiota, jonka myötä pino alkaa kuunnella PCB:lle määrättyä porttia uusia yhteyksiä varten. Kun uusi TCP-yhteys havaitaan, se hyväksytään takaisinkutsun kautta *tcp_accept* -funktiolla. Yhteyden sulkeminen tapahtuu sekä asiakas- että palvelinsovelluksille samalla tavalla. Normaali yhteyden sulkemien tehdään *tcp_close* -funktiolla, ja vikatilanteissa se voidaan lopettaa *tcp_abort* -funktiolla. Vikatilanteita varten on määritettävä oma takaisinkutsu *tcp_err* -funktion avulla.

Datan lähettäminen ja vastaanottaminen tehdään asiakas- ja palvelinsovelluksissa samalla tavalla. Vastaanottamista varten on määritettävä takaisinkutsu *tcp_recv* -funktiolla. Pino kutsuu takaisinkutsufunktiota jokaiselle saapuneelle segmentille, ja takaisinkutsun on kuitattava prosessoitu data *tcp_recvd* -funktiolla. Lähettämistä varten voidaan määrittää oma takaisinkutsu TCP:n kuittauksia (ACK) varten *tcp_sent* -funktiolla. Dataa kirjoitetaan lähetyspuskuriin *tcp_write* -funktiolla. Pino

Taulukko 12: LwIP:n tarjoama rajapinta TCP-protokollalle. [37]

	Funktio	Kuvaus
Yhteyden avaaminen	tcp_new	Luo uuden TCP-PCB:n
	tcp_bind	Kytkee TCP-PCB:n paikalliseen IP-osoitteeseen ja porttiin
	tcp_listen	Aloittaa PCB:n kuuntelemisen
	tcp_accept	Määrittää takaisinkutsufunktion uutta saapuvaa TCP-yhteyttä varten
	tcp_accepted	Kertoo pinolle, että saapuva TCP-yhteys on hyväksytty
	tcp_connect	Avaa TCP-yhteyden
Lähettäminen	tcp_write	Siirtää dataa pinon lähetyspuskuriin odottamaan lähetystä
	tcp_sent	Määrittää takaisinkutsufunktion, jota pino kutsuu, kun vastaanottaja on kuitannut saamansa datan
	tcp_output	Pakottaa pinon lähettämään lähetyspuskurissa olevan datan
Vastaanottaminen	tcp_recv	Määrittää takaisinkutsufunktion saapuvalle datalle
	tcp_recvd	Kertoo pinolle, että saapunut data on prosessoitu
Kiertokysely	tcp_poll	Määrittää takaisinkutsufunktion, jota pino kutsuu säännöllisesti
Yhteyden sulkeminen	tcp_close	Sulkee TCP-yhteyden
	tcp_err	Määrittää takaisinkutsufunktion yhteyden virhetilanteita varten
	tcp_abort	Lopettaa TCP-yhteyden

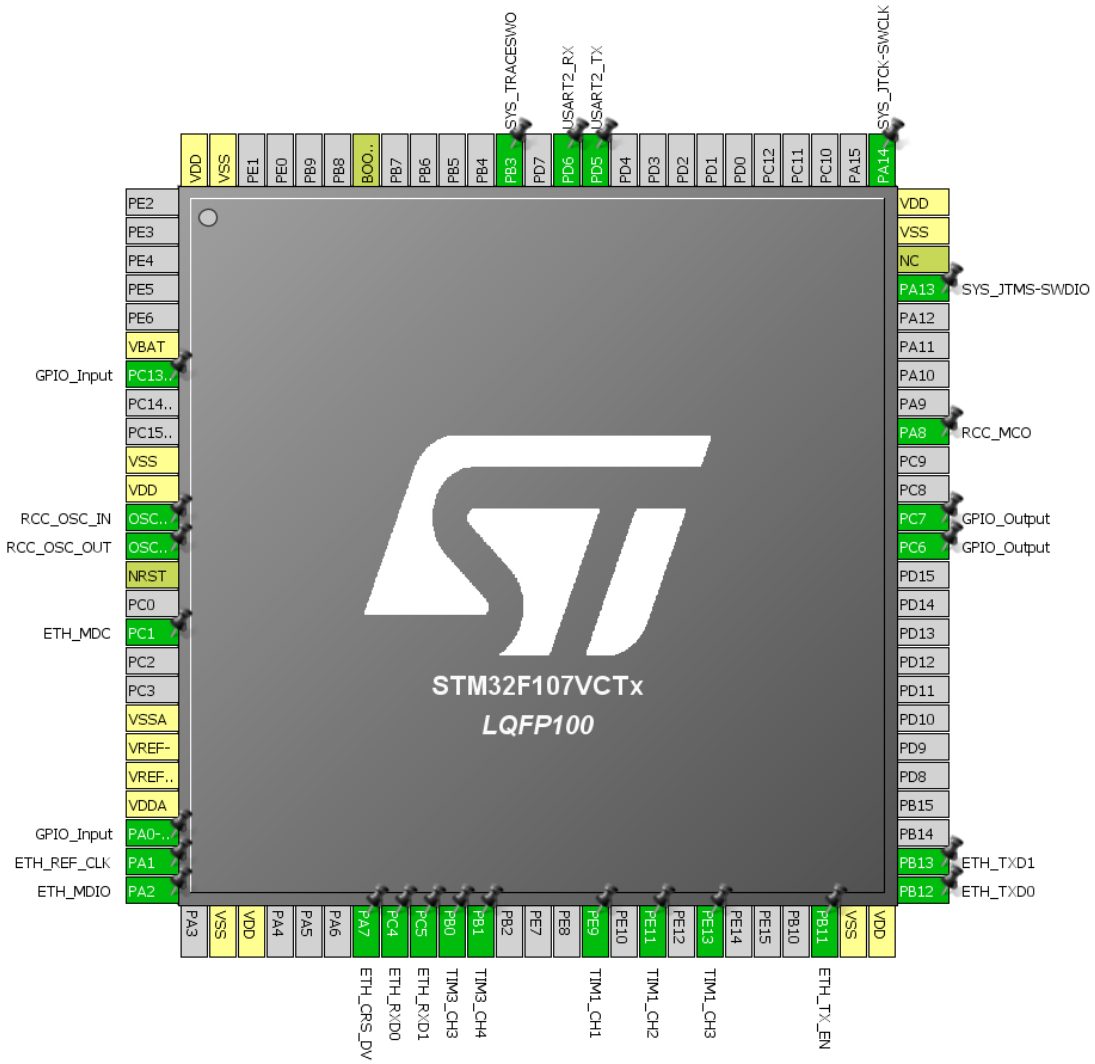
lähettää sen puskurista automaattisesti eteenpäin, mutta tarvittaessa puskuri voidaan pakottaa lähetettäväksi *tcp_ output* -funktiolla. Lisäksi sovellukselle voidaan määrittää kiertokyselyä varten takaisinkutsu, jota pino kutsuu säännöllisin väliajoin. Tämä tehdään funktiolla *tcp_ poll*. [35, 37]

6.4 Ohjainlaitteen sähköinen kytkentä

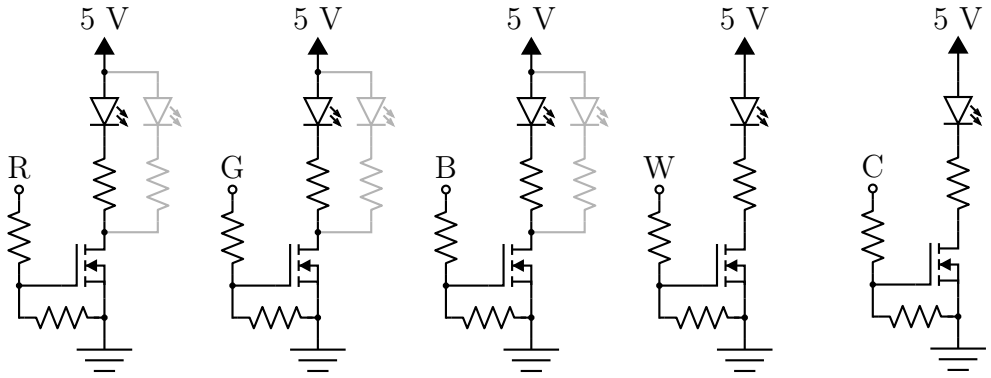
Jotta kehitysalusta säilyy helposti muokattavana, sen prototyyppialueelle kiinnitettiin ainoastaan liittimet, joihin voidaan kytkeä erilaisia lisäkortteja. Vaikka kehitysalustan prototyyppialueen merkinnät muistuttavat mikrokontrollerin pinneistä käytettyjä nimityksiä, ne eivät tarkoita samoja asioita. Selvyyden vuoksi tässä työssä käytetään ainoastaan mikrokontrollerin pinnejä vastaavia nimityksiä.

Kuvassa 23 on esitetty ohjainlaitteessa käytetyt mikrokontrollerin signaalit. ETH-etuliitteellä merkityt pinnit ovat kontrollerin ja PHY-piirin välisen RMII-väylän käytössä. Kontrolleri tuottaa lisäksi väylän vaatiman 50 MHz kellosignaalin RCC_MCO -nimellä varustettuun pinniin. Kehitysalustan kide taas on kytketty RCC_OSC-etuliitteellä merkittyjen pinnien väliin. Myös JTAG-portti otettiin ohjelmointia ja vianetsintää varten käyttöön. Sen varaamat pinnit on merkattu SYS-etuliitteellä. Pinneihin PD5 ja PD6 on kytketty sarjaportti, jota käytettiin laitteen tilan valvontaan. Kehitysalustan merkkivalot ovat lähdöiksi määritetyissä kontrollerin pinneissä PC6 ja PC7 ja kehitysalustan painonapit tuloiksi määritetyissä pinneissä PA0 ja PC13.

Ohjainlaite säätää LEDien kirkkautta pulssinleveysmodulaation, eli PWM:n avulla. PWM-signaalit tuotettiin kuvan 23 mukaisesti mikrokontrollerin ajastinmoduuleilla TIM1 ja TIM3 ja ne kytkettiin ohjelmallisesti kontrollerin pinneihin PB0, PB1,



Kuva 23: Kuvakaappaus STM32CubeMX-ohjelmasta. Kuvasta selviää mikrokontrollerin pinnijärjestys.



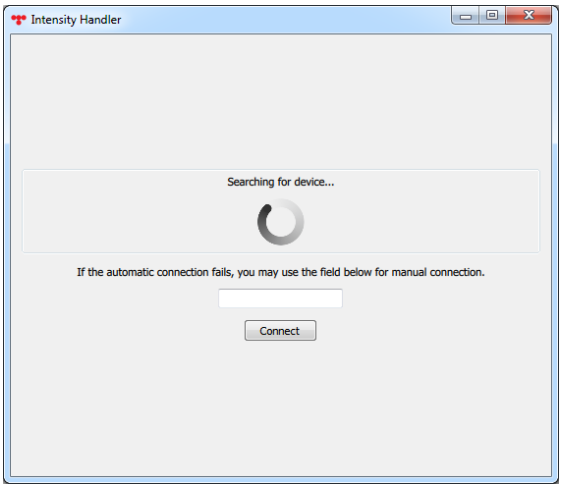
Kuva 24: Ledien kirkkaudensäätöön käytetty kytkentä. Kanavista on käytetty englanninkielisten värien mukaisia lyhenteitä R (punainen), G (vihreä), B (sininen), W (lämmin) ja C (kylmä). Värillisten kanavien yhteyteen harmaalla merkatut haarat vastaavat yhtä kolmiväristä lediä.

PE9, PE11 ja PE13. Mikrokontrollerin tuottamat signaalit vahvistettiin kuvassa 24 esitetyllä lähtöasteella LEDeille. Punaiseen, siniseen ja vihreään kanavaan kytkettiin selkeyden vuoksi sekä yksittäiset, kutakin kanavaa vastaavat värit, että yksi kolmivärinen LED.

6.5 Järjestelmän hallitseminen

Ohjainlaitteen hallitsemiseen kehitettiin Qt-pohjainen tietokoneohjelma. Ohjelman tekninen toteuttaminen rajattiin työn ulkopuolelle, joten tässä yhteydessä esitellään vain ohjelmiston käyttöliittymä ja sen keskeisimmät toiminnot. Jotta ohjainlaite toimisi käytössä olevasta verkosta riippumatta, se käyttää sekä DHCP- että APIPA-menetelmiä oman IP-osoitteensa määrittämiseen. Dynaamisesta osoitteesta johtuen asiakasohjelmiston on selvitettävä laitteen IP-osoite. Siksi ohjelma alkaa käynnistyttyään lähettää tiettyä UDP-sähkettä verkon monilähetysosoitteeseen. Kun ohjainlaite vastaanottaa kyseisen sähkeen, se lähettää ohjelmalle vastauksen, josta selviää valaistuksen senhetkinen tila. Tila on välitettävä, jotta ohjelma voi synkronoida omat parametrinsa vallitsevaan tilanteeseen. Hallintaan käytettävä tietokone ei joissakin tilanteissa onnistunut lähettämään UDP-sähkeitä monilähetysosoitteeseen. Sama tilanne toistui projektin aikana muutamia kertoja, ja tietokoneen uudelleenkäynnistäminen korjasi kaikilla kerroilla ongelman. Jotta järjestelmä olisi edes jollakin tasolla käytettävissä kyseisen ongelman aikana, aloitusnäkymään lisättiin mahdollisuus määrittää ohjattavan laitteiston osoite manuaalisesti.

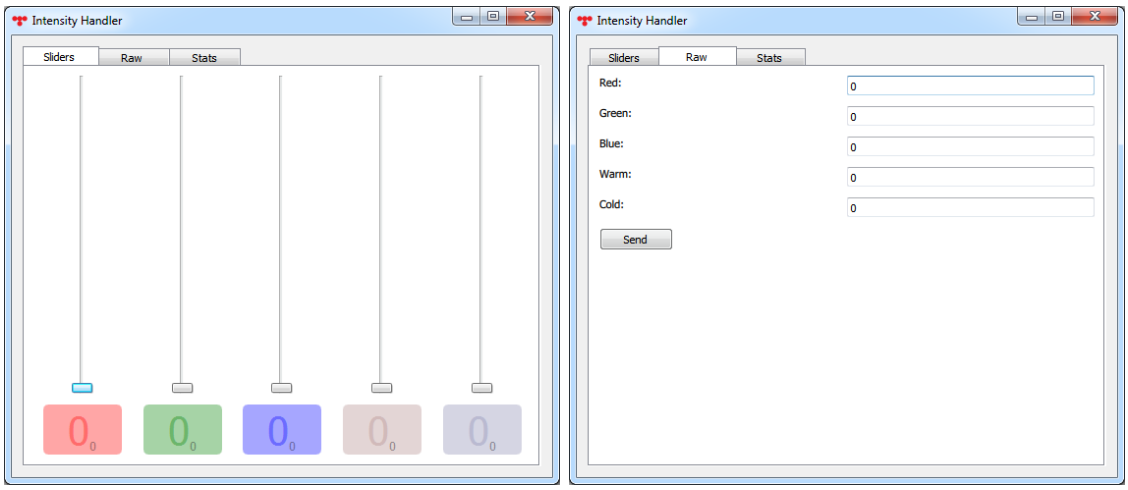
Asiakasohjelma tunnistaa ohjainlaitteen lähettämän vastauksen ja avaa TCP-yhteyden, jota käytetään valojen säätämiseen. Kun yhteys on avattu, ohjainlaite alkaa lähettää säännöllisin väliajoin signaalia, josta asiakasohjelma tietää yhteyden olevan yhä toiminnassa. Mikäli yhteys ohjainlaitteeseen katoaa, ohjelma palaa takaisin kuvassa 25 esitettyyn aloitusnäkymään ja alkaa etsiä laitetta uudelleen UDP-sähkeiden avulla. Periaatteessa sama toiminnallisuus voitaisiin toteuttaa hyödyntämällä TCP-



Kuva 25: Asiakasohjelman aloitusnäkymä. Tähän näkymään palataan myös, mikäli yhteys ohjainlaitteeseen jostakin syystä katoaa.

yhteyden kuittausviestejä, mutta näin ei voitu toimia asiakasohjelman rajoitteiden vuoksi. Kun järjestelmän hallintaan käytettävä TCP-yhteys on avattu, asiakasohjelma avaa kuvassa 26 esitetyn käyttöliittymän, jonka avulla kunkin kanavan taso voidaan määrittää joko liukusäätimillä, tai syöttämällä ohjelmalle lukuarvoja. Liukusäädinten avulla tasojen määrittäminen on havainnollisempaa, mutta tarkkojen arvojen määrittäminen numeroita syöttämällä on etenkin pienellä näytöllä helpompaa.

Järjestelmän ohjaamiseen kehitettiin sovellustason protokolla, jonka komennot muodostuvat kolmen oktetin ryhmistä. Kunkin komennon ensimmäinen oktetti määrittää, minkä kanavan tasoa ollaan säätämässä. Tämän jälkeen kyseisen kanavan taso määritetään kahdella oktetilla. Koska TCP huolehtii yhteydellisenä protokollana seg-



(a) (b)

Kuva 26: Valojen hallintaan kehitetty käyttöliittymä. Tasoja voidaan säätää joko liukusäätimistä (a) tai syöttämällä kutakin kanavaa vastaavat lukuarvot (b).

menttien numeroinnista ja tarkisteista automaattisesti, niitä ei toteutettu uudelleen sovellustason protokollaan. TCP:n käyttämisessä on kuitenkin myös huonoja puolia: se ei takaa, että segmentit saapuvat perille samanlaisina, kuin ne on sovelluksesta lähetetty. Tämä ilmeni käytännössä siten, että käyttöjärjestelmä, jonka varassa hallintaan käytettävä sovellusta käytettiin, pyrki optimoimaan verkkoliikennettä keräämällä useampia segmenttejä omiin puskureihinsa ja yhdistämään ne suuremmiksi segmenteiksi ennen lähettämistä. Puskuroinnin seurauksena datavirta ei ollut jatkuvaa, vaan purskemaista, mikä näkyi valossa askelmaisina muutoksina. Ilmiötä kompensoitiin muuttamalla ohjainlaitteen toimintaperiaatetta siten, että väylältä saapuvia tasoja ei asetettu suoraan kanavien tasoiksi, vaan tasoja ajettiin pehmeästi kohti väylän määrittämiä arvoja. Hyvin nopealla lähetysvälillä havaittiin myös, että osa ohjelmiston lähettämistä segmenteistä saattaa todellisuudessa pirstoutua kahteen peräkkäiseen segmenttiin. Tämä tarkoittaa, että dataa vastaanotettaessa on varmistuttava, että komento on saapunut kokonaisuudessaan perille ennen sen prosessointia. Lisäksi pirstoutumista pyrittiin vähentämään harventamalla segmenttien lähetysväliä.

Koska yksi työn tavoitteista oli kokemuksen ja tiedon hankkiminen, TCP-yhteys korvattiin väliaikaisesti yhteydettömällä UDP-protokollalla. Sovellustason toteutus pidettiin ennallaan. UDP:n käyttäminen korjasi molemmat TCP:n kanssa havaitut ongelmat, sillä UDP-sähkeitä ei puskuroida tai yhdistellä samalla tavalla kuin TCP:n segmenttejä. Teknisesti ottaen myös UDP-sähke voi pirstoutua useampaan IP-pakettiin, mutta tällöin sähkeen uudelleen kokoaminen on alempien kerrosten tehtävä. Sähkeiden ollessa pienikokoisia pirstoutuminen on myös epätodennäköistä. UDP:ta käytettäessä väylältä saadut arvot voitiin asettaa suoraan kunkin kanavan tasoiksi, eikä valossa ollut nähtävissä askelmaisuutta. Menetelmän huonona puolena kuitenkin on se, että yhteydettömän UDP-protokollan sähkeitä ei kuitata, eikä niiden taata saapuvan perille samassa järjestyksessä, kuin ne on lähetetty. Kuittaustoiminto ja sähkeiden järjestäminen voitaisiin toki toteuttaa sovelluskerroksessa. UDP:n testaamista ei kuitenkaan jatkettu, koska tavoitteena oli toteuttaa kommunikointi nimenomaan TCP-protokollan avulla. Lisäksi on mahdollista, että verkon ruuhkautuessa valotasojen askelmaisuus alkaisi näkyä myös UDP:ta käytettäessä.

6.6 Tulosten hyödyntäminen lopputuotteissa

Vaikka suunniteltu ohjainlaite täyttää sille asetetut tavoitteet, se ei sellaisenaan sovellu lopputuotteeksi. Todellisessa ympäristössä laitteita tulee olemaan useampia, ja niitä on voitava hallita yksilöinä. Vaikka jokaiselle laitteelle voidaan osoittaa DHCP-protokollan avulla yksilöllinen IP-osoite, laitteita ei voida helposti erottaa toisistaan. Ongelma voidaan kiertää varustamalla laitteet jollakin ulkoisella tunnisteella, esimerkiksi koodausliittimellä tai dip-kytkimillä. Vaikka ratkaisu on toimiva, sillä on huonojakin puolia. Koodausliittimet ovat omia tuotteitaan, jotka vaativat tilaa niin laitteesta kuin varastostakin. Mekaaniset ratkaisut ovat myös luotettavuuden kannalta ongelmallisia. Osoitteet voitaisiin vaihtoehtoisesti määrittää staattisesti laitteiden käyttöönoton yhteydessä. Näin päästäisiin ylimääräisistä mekaanisista

osista eroon, mutta laitteen käyttöönotto ja vikaantuneen laitteen korvaaminen monimutkaistuvat. Asiakkaan kannalta helpointa ja myös kustannuksiltaan edullisinta on, että laiteelle ei käyttöönoton yhteydessä tarvitse tehdä mitään. Tämä on mahdollista toteuttaa joidenkin teollisten Ethernet-ratkaisujen, kuten PROFINETin, avulla. Tällöin on kuitenkin kyse yksittäisten laitteiden sijaan koko tietoverkon laajuisista ratkaisuista. Muilta osin niin TCP/IP kuin kaikki tässä työssä käsitellyt teolliset Ethernet-ratkaisutkin soveltuvat valaisinjärjestelmän hallitsemiseen.

TCP/IP-protokollaperhettä käytettäessä tiedonsiirtoon käytettävä protokolla tulee valita käyttötarkoituksen mukaan. TCP on protokollana kehittyneempi ja huomattavasti luotettavampi kuin UDP, jolloin se soveltuukin esimerkiksi laitteen asetuksia säätävään rajapintaan hyvin. UDP puolestaan sopii paremmin reaaliaikaiseen tiedonsiirtoon. Sen TCP:tä heikompi luotettavuus on huomioitava sovelluskerroksessa, joko toteuttamalla TCP:n kaltaisia ominaisuuksia, tai määrittelemällä koko protokolla siten, etteivät yksittäisten sähkeiden katoaminen tai virheellinen saapumisjärjestys vaikuta järjestelmän toimintaan.

LwIP-pino toimi kehityksen aikana moitteettomasti, ja se oli STM32CubeMX-työkalun avulla helppo ottaa käyttöön. Pinon lisenssiehdot sallivat kaupallisen käytön, eikä sen käytöstä tarvitse maksaa kaupallisissakaan tuotteissa. Pino on kuitenkin heikosti dokumentoitu. Sen virallinen dokumentaatio on varsin suppea, ja sinänsä melko hyvin kommentoitujen lähdekoodien joukosta löytyy myös ominaisuuksia, joita ei dokumentaatiossa ole kuvattu lainkaan. Ilman ST Microelectronicsin tekemää pohjatyötä pinon käyttöönotto olisi ollut heikon dokumentaation vuoksi merkittävästi hankalampaa. LwIP:lle ei myöskään ole saatavilla kaupalliseen käyttöön soveltuvaa tukea, vaan ainoastaan yhteisöpohjaisia sivustoja. Joiltakin sivustoilta löytyy myös selkeämmin jäsenneltyjä versioita dokumentaatiosta sekä esimerkkejä pinon käytöstä. Näiden yhteisöjen tekemää työtä väheksymättä on todettava, etteivät yhteisöpohjaiset sivustot sovellu kaupalliseen käyttöön, jossa tuotteiden ylläpitoajat ovat pitkiä. Myös dokumentaation puutteesta johtuvat viivästykset saattavat aiheuttaa suuria kustannuksia. Pinon heikon dokumentaation ja tuen vuoksi sen käyttämistä monimutkaisissa järjestelmissä ei voida suositella.

Yleisellä tasolla voidaan sanoa, että Ethernet soveltuu valaisinjärjestelmän hallitsemiseen siinä, missä muutkin kenttäväylät. Tarvittava datamäärä on hyvin pientä, eikä valaisinjärjestelmä itsessään vaadi Ethernetin kaltaista siirtokapasiteettia. Digitaalisiin ohjaustuloihin verrattuna kaksisuuntaiset väylät mahdollistavat esimerkiksi ohjelmistopäivitysten tekemisen sekä järjestelmän diagnostiikkatiedon keräämisen. Pidemmälle kehitettynä järjestelmä voisi esimerkiksi arvioida valaisinten jäljellä olevaa elinaikaa ja ilmoittaa, kun valonlähteet on aika vaihtaa.

7 Yhteenveto

Valaistusalan kehittyminen on uudistanut paitsi valonlähteitä, myös niiden ohjausjärjestelmiä merkittävästi. Julkisessa liikenteessä valaistuksen kuluttaman energian vähentäminen ei ole yhtä tärkeää, kuin alalla yleensä, vaan keskeisiä tekijöitä ovat valonlähteiden elinikä, diagnostiikka ja matkustusmukavuus. Valaisinjärjestelmän kannalta ei ole olennaista minkä kenttäväylän kautta sitä ohjataan, koska tarvittavat datamäärät ovat pieniä. Ethernetin avulla hallitseminen helpottaa kuitenkin valaisinjärjestelmän integrointia muihin järjestelmiin kohteissa, joissa Ethernet on muussakin käytössä.

Ethernet sijoittuu OSI-viitekehyksen kahdelle alimmalle tasolle, eli fyysiselle kerrokselle ja siirtokerrokselle. Siitä on käytössä useita eri versioita, jotka poikkeavat toisistaan pääasiassa siirtokapasiteetin ja käytettävän siirtotien osalta. Yleisimmin käytetty siirtotie on parikaapeli. Sen yleisyys johtuu kaapelin edullisuudesta sekä sen käsittelyn helppoudesta. Yleisimmin käytetty liitin on RJ45-modulaariliitin, mutta haastavissa ympäristöissä voidaan käyttää myös M12-liittimiä. Myös kaapeleista on olemassa eri versioita. Kaikkiin uusiin asennuksiin suositellaan vähintään Cat5e-luokan kaapeleita verkon nopeudesta riippumatta. Tyypillinen toteutus koostuu neljästä komponentista, jotka ovat MAC-piiri, PHY-piiri, edellä mainittujen piirien välinen rajapinta sekä erotusmuuntaja. Piirien välillä käytettävä rajapinta riippuu ensisijaisesti verkon nopeudesta. Tämän lisäksi on valittava käytetäänkö IEEE:n määrittelemää rajapintaa, vai vähempään signaalimäärään perustuvaa pelkistettyä RMII-rajapintaa. Siirtokerroksella voidaan Ethernetin historiasta johtuen käyttää kahta erityyppistä kehystä: alkuperäistä Ethernet II -tyypin kehystä, tai IEEE:n määrittelemää kehystä. Kehykset on helppo erottaa toisistaan, ja verkossa voidaankin käyttää samanaikaisesti kumpaakin kehystyyppiä.

TCP/IP:n määrittelemät protokollat ovat Internetin myötä selkeästi yleisimmin käytettyjä protokollia verkkokerroksella ja sitä ylemillä tasoilla. Se määrittelee oman viitekehyksensä, jossa OSI:n kolme ylintä kerrosta on yhdistetty sovelluskerrokseksi. Verkkokerroksella käytetään IP-protokollaa ja kuljetuskerroksella TCP- ja UDP-protokollia, soveluskerroksen tarjotessa korkeamman tason palveluita. Osa teollisista Ethernet-ratkaisuista perustuu TCP/IP-protokollaperheeseen, kun taas esimerkiksi EtherCAT määrittelee verkkokerroksesta alkaen omat protokollansa. Yksinkertaisimmat teolliset ratkaisut pohjautuvat tyypillisesti johonkin vanhaan sarjaliikenneprotokollaan, jotka on päivitetty käyttämään siirtotienään Ethernetiä. Tämäntyyppiset ratkaisut ovat toimivia, mutta eivät yleensä hyödynnä koko Ethernetin tarjoamaa potentiaalia. Monimutkaisemmat ratkaisut taas tarjoavat työkaluja mm. osoitteiden tehokkaaseen määrittämiseen ja vasteaikojen minimoimiseen. Teolliset Ethernet-verkot noudattavat tyypillisesti vuorosuuntaista 100Base-TX -standardia juuri vasteaikojen ennakoitavuuden vuoksi.

Työn aikana kokeiltiin muutamaa eri STM32-mikrokontrollereille soveltuvaa kehitysympäristöä sekä lwIP-pinoa. Kehitysympäristöistä valittiin käyttöön Atollic TrueSTUDIO. Sen etuihin kuuluivat helppokäyttöisyys, hyvin toimiva integraatio

ST Microelectronicsin tarjoamiin työkaluihin sekä ilmaisversiossakin rajoittamaton ohjelman koko. Myöhempää käyttöä ajatellen TrueSTUDIO:n puolesta puhuvat myös maksullisessa versiossa olevat lähdekoodin analysointiin ja ohjelman testaukseen tarkoitetut työkalut, vaikka niitä ei tämän työn puitteissa voitukaan testata. Joihinkin kehitysympäristöistä voitiin perehtyä vain pintapuolisesti, koska kehitettävää ohjelmaa ei voitu kääntää niillä käytettävissä olleiden versioiden rajoitteiden vuoksi. Niiden käyttöliittymistä jäi kuitenkin lyhyelläkin kokemuksella varsin hankala kuva, joskin vaikutelma käyttöliittymästä on aina subjektiivinen. LwIP-pino toimi työn aikana moitteettomasti, eikä pinosta johtuvia ongelmia havaittu. Sen ei kuitenkaan katsottu soveltuvan teolliseen käyttöön pinon heikon dokumentaation sekä tuen vuoksi.

Soveltavassa osuudessa kehitettiin Olimex STM32-P107 -kehitysalustaan perustuva viittä eri kanavaa hallitseva ohjainlaite ja sen ohjaamiseen soveltuva tietokoneohjelma. Laite ei sellaisenaan sovellu asiakkaalle myytäväksi tuotteeksi, vaan profiloituu esimerkiksi valaisinten esittelykäyttöön soveltuvaksi laitteeksi. Sen suurin puute on IP-osoitteen päätteleminen tilanteessa, jossa useita identtisiä laitteita on kytkettynä samaan verkkoon. Laitteille saadaan kyllä yksilöllinen osoite DHCP-protokollan avulla, mutta tietyn laitteen IP-osoitteen selvittämistä pelkän TCP/IP-protokollaperheen avulla ei onnistuttu työssä ratkaisemaan. TCP-protokollan havaittiin soveltuvan lähinnä tiedonsiirtoon, joka ei ole erityisen aikakriittistä. Laitteen asetusten säätäminen tai ennalta määrättyjen valotasojen valitseminen ovat esimerkkejä TCP:lle soveltuvista tehtävistä. UDP-protokollan avulla sen sijaan voidaan lähettää huomattavasti aikakriittisempää tietoa ja tehdä monilähetyksiä. Se ei kuitenkaan ole yhtä luotettava protokolla, kuin TCP.

Vaikka kehitetty laitteisto ei ole valmis tuote, työn tuloksia voidaan hyödyntää tulevissa laitteissa. Koska Teknowaren tarkoituksena ei ole toimittaa koko junan tietoverkkoa, käytettävistä rajapinnoista on sovittava tapauskohtaisesti. Käytännössä määrittävä tekijä on asiakkaan käyttämä verkko ja sen tukemat protokollat. Tulevissa projekteissa asiakkaan verkon asettamia vaatimuksia ja sen tarjoamia mahdollisuuksia voidaan tulkita tämän työn kokemusten valossa, mikä oli toinen työn päätavoitteista. Toinen tavoitteista oli lwIP-pinoon ja kehitystyökaluihin perehtyminen. Myös tämän tavoitteen voidaan katsoa täyttyneen, vaikka kyseistä pinoa ei suositellakaan käytettäväksi.

Viitteet

[1] Bergman, David. *Sustainable Design: A Critical Guide for Architects and Interior, Lighting, and Environmental Designers.* New York, NY, Princeton Architectural Press, 2012. ISBN 978-1-56898-941-9.

[2] Baggini, Angelo, Sumper, Andreas. *Electrical Energy Efficiency: Technologies and Applications.* Chichester, John Wiley & Sons, 2012. ISBN 978-0-470-97551-0

[3] Reynders, Deon, Mackay, Steve, Wright, Edwin. *Practical Industrial Data Communications: Best Practice Techniques.* Oxford, Newnes, 2005. ISBN 0-7506-6395-2.

[4] Uotila, Pekka. *Tietoliikenteen tekniikka.* Jyväskylä, Gummerus Kirjapaino Oy, 2000. ISBN 951-762-694-0.

[5] Dostálek, Libor, Kabelová, Alena. *Understanding TCP/IP.* Birmingham, Packt Publishing, 2006. ISBN 1-904811-71-X.

[6] Edwards, James, Bramante, Richard. *Networking : OSI, TCP/IP, LANs, MANs, WANs, Implementation, Management, and Maintenance.* Indiapolis, IN, Wiley Publishing, 2009.

[7] Reynders, Deon, Wright, Edwin. *Practical TCP/IP and Ethernet Networking.* Oxford, Newnes, 2003. ISBN 07506 58061.

[8] *Registration fees.* IEEE Standards Association. Verkkodokumentti. Saatavilla: `http://standards.ieee.org/develop/regauth/oui/`. Viitattu 18.10.2015.

[9] *Guidelines for Use Organizationally Unique Identifier (OUI) and Company ID (CID).* IEEE Standards Association. Verkkodokumentti. Saatavilla: `https://standards.ieee.org/develop/regauth/tut/eui.pdf`. Viitattu 18.10.2015.

[10] Axelson, Jan. *Embedded Ethernet and Internet Complete.* Madison, WI, Lakeview Research, 2003. ISBN 1-931448-01-9.

[11] Marshall, Perry S. Rinaldi, John S. *Industrial Ethernet.* Research Triangle Park, NC, ISA, 2004. ISBN 1-55617-869-7.

[12] *IEEE Std 802.3-2005 Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical layer Specifications.* IEEE Standards Association, 2005.

[13] *RMII Specification.* RMII Consortium, versio 1.2, 1998. Verkkodokumentti. Saatavilla: `http://ebook.pldworld.com/_eBook/-Telecommunications, Networks-/TCPIP/RMII/rmii_rev12.pdf`. Viitattu: 16.11.2015.

[14] IANA:n verkkosivut. Saatavilla: `www.iana.org`. Viitattu: 5.11.2015.

[15] Blank, Andrew G. *TCP/IP JumpStart.* Alameda, CA, Sybex, 2002. ISBN 0-7821-4101-3.

[16] Modbusin verkkosivu. Saatavilla: `http://modbus.org`. Viitattu: 17.12.2015.

[17] *MODBUS Messaging on TCP/IP Implementation Guide.* Modbus-IDA, versio 1.0b, 2006. Verkkodokumentti. Saatavilla: `http://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf`.

[18] *MODBUS Application Protocol Specification.* Modbus-IDA, Versio 1.1b, 2006. Verkkodokumentti. Saatavilla: `http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf`.

[19] ODVA:n verkkosivu. Saatavilla: `https://www.odva.org/`. Viitattu: 28.12.2015.

[20] *EtherNet/IP Quick Start for Vendors Handbook.* Open DeviceNet Vendor Association, julkaisunumero PUB00213R0, 2008. Verkkodokumentti. Saatavilla: `https://www.odva.org/Portals/0/Library/Publications_Numbered/PUB00213R0_EtherNetIP_Developers_Guide.pdf`. Viitattu: 28.12.2015.

[21] *EtherCAT Slave introduction and Implementation Guide.* EtherCAT Technology Group, dokumenttinumero ETG.2200, versio 2.1.2, 2014. Verkkodokumentti. Saatavilla: `https://www.ethercat.org/download/documents/ETG2200_V2i1i2_G_R_SlaveImplementationGuide.pdf`. Viitattu: 24.1.2016.

[22] *EtherCAT Technology Introduction.* EtherCAT Technology Group, 2012. Verkkodokumentti. Saatavilla: `https://www.ethercat.org/download/documents/EtherCAT_Introduction_EN.pdf`. Viitattu: 24.1.2016.

[23] *Profinet System Description - Technology and Application.* PROFIBUS Nutzerorganisation, 2014. Verkkodokumentti. Saatavilla: `http://www.profibus.com/nc/download/technical-descriptions-books/downloads/profinet-technology-and-application-system-description/download/18676/`. Viitattu: 28.2.2016.

[24] Olimex STM32-P107 -kehitysalustan verkkosivu. Saatavilla: `https://www.olimex.com/Products/ARM/ST/STM32-P107/`. Viitattu: 14.11.2015.

[25] ST-Link/V2 -ohjelmointilaitteen verkkosivu. Saatavilla: `http://www.st.com/web/catalog/tools/FM146/CL1984/SC724/SS1677/PF251168`. Viitattu: 14.11.2015.

[26] *STM32-P107 development board User's manual.* Olimex Ltd., Rev. K, 2015. Verkkodokumentti. Saatavilla: `https://www.olimex.com/Products/ARM/ST/STM32-P107/resources/STM32-P107-MANUAL.pdf`.

[27] IAR Embedded Workbench -kehitysympäristön verkkosivu. Verkkodokumentti. Saatavilla: `https://www.iar.com/iar-embedded-workbench`.

[28] Keil MDK -kehitysympäristön verkkosivu. Verkkodokumentti. Saatavilla: http://www.keil.com/mdk5.

[29] System Workbench for STM32 -kehitysympäristön verkkosivu. Ohjelman lataaminen ja käyttöohjeiden selaaminen vaatii ilmaisen rekisteröinnin. Saatavilla: http://www.openstm32.org. Viitattu: 14.11.2015.

[30] Atollic TrueSTUDIO -kehitysympäristön verkkosivu. Verkkodokumentti. Saatavilla: http://timor.atollic.com/truestudio/.

[31] STM32CubeMX-ohjelman verkkosivu. Saatavilla: http://st.com/stm32cube. Viitattu: 14.11.2015.

[32] STM32 ST-LINK utility -ohjelman verkkosivu. Saatavilla: http://www.st.com/web/en/catalog/tools/PF258168. Viitattu: 14.11.2015.

[33] Packet Sender -ohjelman verkkosivu. Saatavilla: https://packetsender.com. Viitattu: 14.11.2015.

[34] Wireshark-ohjelman verkkosivu. Saatavilla: https://wireshark.org. Viitattu: 14.11.2015.

[35] Dunkels, Adam, Woestenberg, Leon, Simons, Christiaan. *Raw TCP/IP interface for lwIP*. Dokumentti toimitetaan lwIP-pinon mukana. Pino on saatavilla verkossa, osoitteessa http://savannah.nongnu.org/projects/lwip/. Viitattu: 3.12.2015.

[36] *AN3102: lwIP TCP/IP stack demonstration for STM32F107xx connectivity line microcontrollers.* ST Microelectronics, DocID 16620, Rev 1, 2009. Verkkodokumentti. Saatavilla: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/application_note/CD00255062.pdf Viitattu: 14.11.2015.

[37] *UM1713: Developing applications on STM32Cube with LwIP TCP/IP stack.* ST Microelectronics, DocID 025731, Rev 4, 2015. Verkkodokumentti. Saavavilla: http://www.st.com/st-web-ui/static/active/jp/resource/technical/document/user_manual/DM00103685.pdf. Viitattu: 16.11.2015.