Petteri Saarikko

LoRa-pohjainen monihyppyverkko. Case Meshtastic

Tietotekniikan
Pro gradu -tutkielma
5. kesäkuuta 2023

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Kokkolan yliopistokeskus Chydenius

Tekijä: Petteri Saarikko
Yhteystiedot: petteri.saarikko@gmail.com
Puhelinnumero: 040-5078866
Ohjaaja: Ismo Hakala
Työn nimi: LoRa-pohjainen monihyppyverkko. Case Meshtastic
in English: Lora-based multihop network. Case Meshtastic
Työ: Tietotekniikan Pro gradu -tutkielma
Sivumäärä: 53
Tiivistelmä: Esineiden internetin (IoT) taustalta löytyy paljon erityyppisiä protokollia, tiedonsiirtotekniikoita ja vähävirtaisia ratkaisuja, joita sovelletaan verrattain vähän muissa kuin telemetria eli mittaus- ja ohjaustarkoituksissa. IoT-teknologioiden avulla on mahdollista muodostaa viestintäpalveluita, joiden riippuvuus julkisesta infrastruktuurista kuten internetistä voidaan tarvittaessa poistaa kokonaan. Pro gradussa tarkastellaan LoRa-protokollaa ja sen ympärillä tehtyä monimuotoista tutkimusta. LoRa-protokolla tuottaa perustan Mehstastic sovelluskehikolle, joka mahdollistaa yhtäaikaisen tekstiviesti, GPS- ja telemetriatietojen välittämisen ilman julkisen tietoliikenneinfrastruktuurin palveluita. Meshtastic sovelluskehikon avulla muodostettiin esimerkkisovellus todentamaan tätä toimintaa.
Avainsanat: LoRa, LoRaWAN, Mesh, Meshtastic
Abstract: The Internet of Things (IoT) constructs by a variety of protocols, data transfer technologies, and low-power solutions that are applied relatively little outside of telemetry, i.e., measurement and control purposes. With IoT technologies, it is possible to create communication services that can, if necessary, completely eliminate dependency on public infrastructure such as the internet. This thesis examines the LoRa protocol and the diverse research conducted around it. The LoRa protocol provides the foundation for the Meshtastic application framework, which allows for simultaneous text messaging, GPS and telemetry data transmission without the services of public telecommunications infrastructure. With the help of the Meshtastic application framework, a demonstration application was created to verify this operation.
Keywords: LoRa, LoRaWAN, Mesh, Meshtastic

Esipuhe

Pro gradun aihe valikoitui sattumalta ja työ on kirjoitettu ilman toimeksiantajaa perustuen henkilökohtaisiin mielenkiinnonkohteisiin. Tältä osin haluan kiittää ohjaajani professori Ismo Hakalaa, joka on osaltaan mahdollistanut tämän pro gradun syntymisen.

Sanasto

ABP	LoRaWAN tunnistustekniikka (engl. Activation By Personalization)
ADR	Dynaamisesti muuttuva tiedonsiirtonopeus (engl. Adaptive Data Rate)
BW	Kanavataajuus (engl. Bandwith)
Chip	Hajaspektrimodulaation ajallinen osa (engl. LoRaWAN time spread unit)
CR	Virheenkorjauksessa käytettävien bittien määrä (engl. Coding rate)
CSS	Hajaspektrimodulaatio (engl. Chirp spread spectrum)
DR	Siirtonopeus (engl.Data rate)
Duty cycle	Vuoronvaraus (engl. Duty cycle)
FEC	Ennakoiva virheekorjaus (engl. Forward error correction)
GPIO	Fyysinen liittymä jota voidaan ohjelmoida digitaalisesti (engl.general-purpose input/output)
ISM	Teollinen, tieteellinen ja lääketieteellinen (engl. Industrial, scientific and medical)
LoRa	Langaton modulointitekniikka (engl. Wireless modulation for long-range, low-power, low-data-rate)
LoRaWAN	Vähävirtainen, avoin langaton tiedonsiirto tekniikka (engl. LoRa wide area network)
LPWLAN	Vähävirtainen pitkän kantaman verkko (engl. Low-power wide area network)

OTAA	LoRaWAN tunnistustekniikka (engl. Over-The-Air Activation)
SF	Hajautuskerroin (engl. Spreading factor)
ToA	Aika minkä tiedonsiirto varaa kanavataajuudelta (engl. Time on air)
WSN	Langaton sensoriverkko (engl. Wireles sensor network)

Sisällys

1 Johdanto

Esineiden internetin taustalta löytyy paljon erityyppisiä protokollia, tiedonsiirtotekniikoita ja vähävirtaisia ratkaisuja, joita sovelletaan verrattain vähän muissa kuin IoT-käyttötarkoituksissa. IoT-ratkaisuille on ominaista, että ne eivät välttämättä tarvitse toimiakseen ulkopuolista infrastruktuuria tai ainakin toimivat julkisen infrastruktuurin kuten internetin jatkeena. Arkipäiväiset toimemme kuten viestiminen ovat erittäin riippuvaisia julkisen infrastruktuurin palveluista, ja IoT-ratkaisuiden avulla voisi olla mahdollista vähentää tätä riippuvuutta. Vuonna 2023 maailman geopoliittinen tilanne esittäytyy epävarmana ja infrariippumattomien toteutuksien tarkastelulle on tilaa jo tästäkin näkökulmasta.

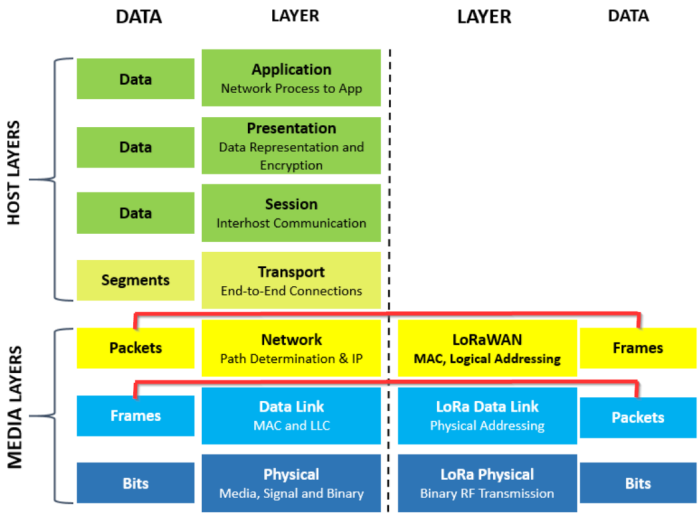
IoT-protokollat ja näihin kytkeytyvät sovellukset mielletään useasti käytettäväksi osana automaatioprosessia kuten mittauksen tai ohjauksen mahdollistajana. IoT-protokollien avulla voidaan kuitenkin muodostaa esimerkiksi tekstiviestipalveluita kuitenkaan poistamatta mittauksen tai ohjauksen mahdollisuutta. Jotta infrastruktuurittomuus voi toteutua tulee verkkojen kyetä dynaamiseen tiedonvälitykseen riippumatta siitä paljonko verkkoon kuuluu reitittäviä tai välittäviä laitteita jollain ajanhetkellä. Mesh-verkko välittää tietoa kaikille siellä oleville laitteille ja voi näin ollen mahdollistaa infrastruktuurittoman ratkaisumallin.

Tutkimuskysymyksiä on kaksi. Minkälaista tutkimusta *LoRa* tai *LoRaWAN* mesh-eli monihyppyverkkojen osalta on tehty ja miten Meshtastic sovelluskehys otetaan käyttöön ja kuinka se soveltuu tekstiviesti- ja mittausverkon palveluiden yhtäaikaiseen tuottamiseen?

Luvussa kaksi esitellään *LoRa* ja *LoRaWAN* tekniikoiden perusteet ja nämä luovat tarvittavan tietopohjan tutkimustulosten tarkastelulle. Luvussa kolme tarkastellaan tutkimustuloksia ja erityyppisiä toteutustapoja mesh-verkkokoille *LoRa* ja *LoRaWAN* tekniikoilla. Luvussa 4 esitellään Meshtastic sovelluskehikko ja sen tuottamat palvelut *LoRa*:n perustuvan mesh-verkon tuottajana. Luvussa 4 esitellään myös hybriditoteutus, missä infrastruktuurittomasta *LoRa* mesh verkosta välitetään tekstiviesti- ja telemetriadataa internettiin *MQTT*-protokollan avulla.

2 LoRa ja LoRaWAN

LoRa on Semtechin omistama fyysisen kerroksen määritelmä, missä kuvataan datan modulointi ja pakettirakenne *LPWLAN*-tiedonsiirrossa [38]. Verkkojulkaisussa [37] kuvataan *LoRa*-moduloinnin perusteita ja on yksi harvoista omistajansa julkaisemista modulaatiota käsittelevistä dokumenteista. *LoRaWAN* on tietoliikenneprotokolla, jonka kehityksen on käynnistänyt *LoRa*-allianssi vuonna 2015. Perustamisen yhteydessä Semtech liittyi allianssin jäseneksi ja luovutti *LoRa*:n käyttöoikeuden tälle. Allianssin tarkoituksena on kehittää avoin kommunikointitapa *LPWLAN*-laitteille, jotka käyttävät virtalähteenään paristoa [24]. *LoRaWAN* käyttää *LoRa*:n tuottamia fyysisen- ja datalinkkikerroksen palveluita ja tuottaa näiden päälle verkkokerroksen (kuva 2.1). Kuvasta 2.1 on hyvä huomioida, että OSI-mallin mukainen kehyksien ja pakettien kerrosjako on päinvastainen *LoRa* ja *LoRaWAN*-kerroksilla. *LoRaWAN* ei voi käyttää fyysisellä kerroksella mitään muuta kuin *LoRa*:n palveluita ja vastaavasti ainoastaan *LoRa*:n käyttö edellyttää jotain sovelluskirjastoa, joka kykenee muodostamaan ja käsittelemään hyötykuorman sisältöä.



Kuva 2.1: *LoRa* ja *LoRaWAN* OSI-malli [38]

2.1 LoRa

Hajaspektrimoduloinnin johdosta *LoRa* sietää hyvin ympäristössä olevia häiriöitä kuten monitiekuuluvuutta, luonnollista kohinaa tai signaalin vaimenemista. Hajaspektrimodulointi yhdessä *LoRa*-radion kanssa mahdollistaa jopa 10 kilometrin kuuluvuuden avoimissa ympäristöissä vähäisellä virrankulutuksella [4]. Thethingsnetwork [44] kertoo verkkotiedoitteessaan 766 kilometrin päätelaiteyhteydestä 25 milliwatin lähtysteholla LoRa:n avulla. Kyseessä on ollut optimoitu testausympäristö, missä päätelaite on kytketty säähavaintopallon ja tämä on mahdollistanut esteettömän yhteyden laitteiden välille.

2.1.1 Modulointi ja fyysisen kerroksen toiminta

CSS (engl. Chirp Spread Spectrum) on hajaspektri modulaatiotekniikka, jonka avulla muodostetaan symboleja tiedonsiirtoa varten. Modulointi perustuu lineaariseen taajuusvaihteluun. Taajuusvaihtelu on jatkumo eli jos modulointitaajuus saavuttaa kanavataajuuden maksimin jatkuu se kanavataajuuden minimistä (kuva 2.2, kohta 1) [45][42]. *Chirp* on moduloinnin yksikkö, joka suomentuu viserrykseksi ja voi olla moduloitu tai moduloimaton taajuusvaihtelu tietyssä aikaikkunassa. Moduloitu viserrys on taajuudeltaan nouseva ja sille on ominaista, että sen aloitus- ja lopetustaajuudet vastaavat toisiaan, (kuva 2.2 kohta 2) [45][42]. Moduloimattomia viserryksiä käytetään tiedonsiirron kättelyyn ja aikasynkronointiin (engl. preamble) kun vastaavasti hyötykuorma on kokonaisuudessaan moduloitua (kuva 2.5) [45][42].

CSS-modulaatio käyttää hajautuskerrointa *SF* viserryksenmuodostuksen määrityksessä. Käytössä on kuusi eri SF-arvoa välillä 7–12, jotka määrittävät symbolinmuodostukseen kulutettavan ajan ja koodattavien bittien määrän. Kuvassa 2.2 voidaan havaita kuinka suurempi SF-arvo vaikuttaa viserryksen taajuuden kulmaan. Esimerkkinä kuvan 2.2 kohdassa 4 olevat viserrykset *SF* 8 ja *SF* 12, jotka ovat kanavataajuustasolla samat, mutta niiden vaihekulma muuttuu pidemmän aikaikkunan takia[45]. Jokaisen SF-arvon kasvattaminen kaksinkertaistaa edellisen aikaikkunan ajan [42]. Eri SF-arvot mahdollistavat datan yhtäaikaisen lähettämisen samassa kanavataajuudessa. Käytössä olevia kanavia on siis kanavataajuudet jaettuna *SF* arvoperusteisesti alikanaviin [38]. Yksittäinen kehys muodostetaan valitulla SF-arvolla eli arvo ei muutu kehyksen sisällä [6].

Muodostettaessa symbolia jakautuu *SF* arvon määrittämä aikaikkuna 2^{SF} osaan (engl. chip) [42]. Esimerkiksi viserryksessä, jonka muodostuksessa on käytetty ha-

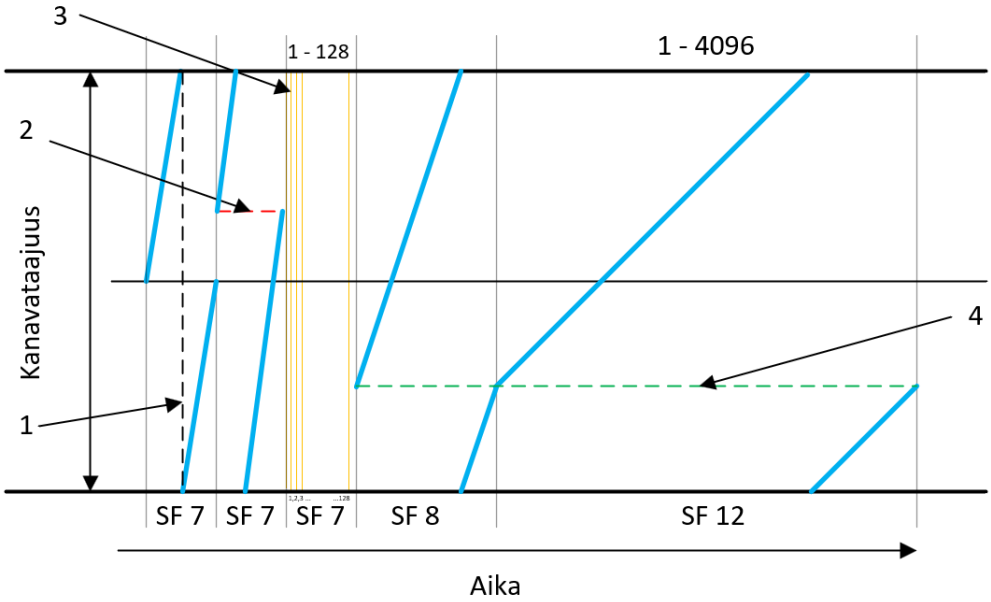
jautusarvoa *SF* 7 on 128 osaa (kuva 2.2 kohta 3). Vastaavasti viserrys, joka on koodattu arvolla *SF* 12 jakaantuu 4096 osaan. Kanavataajuus voidaan ilmaista osissa (engl. chip) kaavalla

$$BW = R_c = chiprate(chips/s).$$

Näin ollen kanavataajuus 125 kHz muuntuu arvoon 125 000 *chips/s*. Tiedon avulla voidaan laskea koodattavien symbolien määrä sekunnissa halutulla *SF* arvolla

$$R_s(symbols/s) = R_c/2^{SF}.$$

Esimerkiksi hajautuskerroin *SF* 7 kanavataajuudella 125 kHz vastaa 977 symbolia sekunnissa (125 000 / 2^7 = 977). Vastaavasti SF-arvolla 12 muodostuu 30,5 symbolia sekunnissa (125 000 / 2^{12} = 30,5). Korkeampi hajautusarvo vähentää siirrettävän datan määrä, koska koodattavien symbolien määrä vähenee suhteessa kuluvaan aikaan. Korkeampi kanavataajuus kaksinkertaistaa muodostuvien symbolien määrän. Verrattaessa aiemman esimerkin hajautuskerrointa *SF* 12 kanavataajuudella 250 kHz saadaan symbolimääräksi 61 symbolia sekunnissa (250 000 / 2^{12} = 61). Suuremman SF-arvon käyttäminen parantaa signaalikuuluvuutta eli verkon kattavuutta ja sietää paremmin luonnollista kohinaa ja muita häiriöitä[45] [42].



Kuva 2.2: Yksittäisiä moduloituja esimerkki viserryksiä

Taulukko 2.1: CR-arvot [32]

CR-arvo	1	2	3	4
Redundantti bittien määrä	1	2	3	4
CR-arvo	$\frac{4}{5}$	$\frac{4}{6}$	$\frac{4}{7}$	$\frac{4}{8}$

2.1.2 CR ja FEC

Julkaisussa [32] kuvataan *CR* (engl. Code Rate) ja *FEC* (engl. Forward error correction) toimintaa seuraavasti. *FEC* on virheenkorjaustekniikka, jota käytetään parantamaan vastaanottajan mahdollisuuksia tulkita saapuvan hyötykuorman sisältöä oikein. *FEC* käyttäminen ei ole pakollista, jolloin kehys muuttuu implisiittiseen muotoon. Mikäli *FEC* on käytössä muuntuu kehys explisiittiseen muotoon (kuva 2.6). Symbolimuodostuksen aikana *FEC* laskee hyötykuormaan pariteettibittejä, joiden avulla vastaanottaja voi tarkistaa saapuneen datan eheyden. Mikäli pariteettilaskenta ei täsmää voidaan hyötykuorman sisältö laskea vastaanottajan toimesta pariteettitiedon avulla. *CR*-arvo määrittää ennakoivan virheenkorjauksen pariteettibittien määrän ja arvot voivat olla jotain 0 – 4 väliltä. Arvon ollessa 0 ei ennakoivaa virheenkorjausta käytetä määrityksen toimiessa muuten taulukon 2.1 mukaisesti. Taulukossa 2.1 esitetään murtoluku perusteinen esitystapa *CR*-arvon ilmaisemiselle. Lähteestä riippuen molemmat esitystavat ovat käytössä. Ennakoivan virheenkorjauksen käyttäminen pienentää hyötykuorman dataosuutta, koska pariteettibitit lasketaan osaksi sitä. Kehys on kuvattu tarkemmin alaluvussa 2.2.7.

.

2.1.3 Siirtonopeuksien laskeminen

Edellä on esitetty tarvittavat muuttujatiedot tiedonsiirtonopeuksien laskemista varten. *DR* (engl. Data rate) tai toisin ilmaistuna R_b = bittiä per sekunti voidaan laskea kaavalla

$$R_b(bits/sec) = SF \times \frac{BW}{2^{SF}} \times \frac{4}{(4+CR)}.$$

Esimerkiksi kanavataajuuden ollessa 125kHz, hajautuskertoimen *SF* ollessa 7 ja käytettäesssä virheenkorjausarvoa *CR* 1 siirtonopeus on $R_b = 7 \times (125000/2^7) \times (4/(4+1) = 5.5kbits/s$. Aiemmin esitetty väittämä, kanavataajuuden kaksinkertaistami-

nen kaksinkertaistaa siirrettyjen symbolien määrän voidaan todentaa seuraavasti. $R_b = 7 \times (250000/2^7) \times (4/(4+1) = 10.9 kbits/s$.

2.1.4 Käyttösuhde

Artikkelissa [6] tarkastellaan eri hajautuskertoimien vaikutusta käyttösuhteeseen (engl. duty cycle). Relaatio suhteessa käytössä olevaan kanavataajuuteen *BW* ja hajautuskertoimeen *SF* on merkittävä tarkasteltaessa lähetysaikaa *ToA* (engl. Time on Air), jonka lähetys varaa kanavalta tiedonsiirron aikana. Kaksi lähetystä, jotka ovat hyötykuormaltaan samankokoiset ja käyttävät samaa kanavataajuutta, mutta ovat lähetetty eri SF-arvon perusteella poikkeavat merkittävästi lähetysajan osalta toisistaan. Hyötykuorman ollessa 50 tavua, kanavataajuuden ollessa 125 kHz ja SF-arvon ollessa 7 lähetysaika eli *ToA* on 0.113 sekuntia. Vastaavasti SF-arvon ollessa 12 lähetysaika on 2.62 sekuntia.

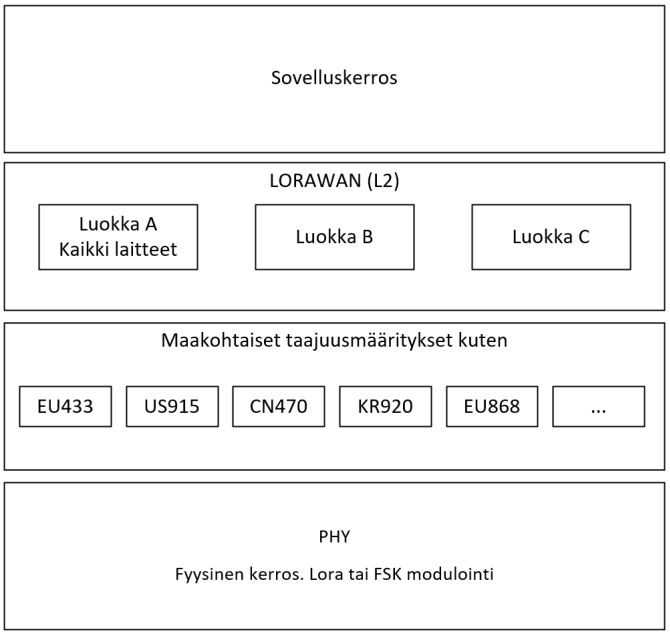
Taulukossa 2.2 ja 2.3 ilmoitetaan käyttösuhde joka täytyy olla vähemmän kuin 1 %. Tällä viitataan aikaan, jonka yksittäinen päätelaite saa maksimissaan käyttää lähettäessään kanavalle dataa. Artikkelissa [34] kuvataan laskentaperiaate seuraavasti. Prosenttiarvo on lähetyksen kuluneen ajan yhteenlaskettu kumulatiivinen summa yhden tunnin ajalta, eli yhden tunnin seurantajakson aikana yksittäinen päätelaite saa lähettää maksimissaan 36 sekuntia. Edellä mainittua yhden tunnin aikaikkunaa lasketaan ensimmäisestä päätelaitteen lähetyksestä eteenpäin. Käyttösuhde on kanavataajuusperusteinen, eli yksi päätelaite voi lähettää usealla eri kanavataajuudella yhtäaikaisesti ja jokaista kanavataajuutta tarkastellaan omana arvonaan.

2.2 LoRaWAN

LoRaWAN toimintaperiaate voidaan esittää kuvan 2.3 mukaisella tavalla. Pohjalla on datan modulointi ja paketointi. Paketoitua dataa kuljetetaan maakohtaisen ISM-taajuusmäärityksen määrittämällä tavalla päätelaitteen ja yhdyskäytävän välillä. Päätelaitteet ja yhdyskäytävät kuuluvat johonkin ennalta määriteltyyn päätelaiteluokkaan ja sovellus keskustelee edellä olevien kautta sovelluspalvelun kanssa [25].

LoRaWAN-spesifikaatio [25] kuvaa päätelaiteluokituksen seuraavasti. Verkossa olevat päätelaitteet kuuluvat johonkin L2-luokkaan (kuva 2.3) laitteen kyvykkyyden ja sovellustarpeiden mukaisesti. Kaikki päätelaitteet toteuttavat vähintään A-

luokan ominaisuudet. Päätelaitteen luokitusta voidaan muuttaa, joko määrittämällä se sovelluskerroksella tai suoraan päätelaitteella. A-luokan päätelaitteille on ominaista kaksisuuntainen kommunikaatio, jossa jokaista lähetystä seuraa kaksi vastaanottoaikaikkunaa. A-luokan laitteet käynnistävät kommunikaation itsenäisesti ja koska päätelaite ei kuuntele verkossa olevaa liikennettä voidaan päätelaitteen virrankulutusta kontrolloida sovelluskerroksen kautta. A-luokan laitteilla verkko, yhdyskäytävä tai sovellus joutuu odottamaan päätelaitteen aloittamaa kommunikointia mahdollisissa datanvälitys tilanteissa kohti päätelaitetta. B-luokan päätelaitteet omaavat A-luokan ominaisuudet, mutta kuuntelevat verkkoa ajastettavien aikaikkunoiden avulla. Yhdyskäytävä välittää verkkopalvelusta aikatietokehyksen, minkä perusteella päätelaite tietää olla valveilla datan vastaanottoa varten. Aikakehystietoa hallinnoi verkkopalvelin. C-luokan päätelaitteet omaavat A- ja B-luokan ominaisuudet ja ovat jatkuvasti päällä. Tämä mahdollistaa pienemmät viiveet lähettämiselle ja vastaanottamiselle. C-luokan laitteet ovat normaalisti yhdyskäytäviä tai muita kiinteänvirtalähteen omaavia laitteita. Yhdyskäytävillä on myös useampi radiomoduuli yhtäaikaista tietoliikennöintiä varten.



Kuva 2.3: LoraWan kerroskuvana [25]

2.2.1 Kanavataajuudet

LoRaWAN toimii lisensoimattomalla alle gigahertsin ISM-radiotaajuudella. Jokaisella maalla on oma taajuusregulaationsa, joka määrittää sallitut taajuusalueet ja muut aluekohtaisert määritykset. Euroopassa on yleisesti käytössä 868MHz taajuus ja esimerkiksi USA ja Australia käyttää 915MHz taajuutta. Vaikka taajuusalueet ovat joissain tapauksissa samoja kanavajakoperusteet saattavat poiketa eri maiden välillä. Esimerkiksi USA ja Australia käyttävät samaa 915MHz taajuusaluetta, mutta lähetys- ja vastaanottokanavien taajuusaluekäytännöt eroavat toisistaan [39]. *LoRaWAN* yhteenliittymän maakohtaisessa dokumentaatiossa Suomi luokitellaan EU433 ja EU868 säännösten alaisuuteen, mikä tarkoittaa 433,05 – 434,79MHz ja 863 – 873MHz välisiä taajuusalueita [26]. Verkko-operaattori voi jakaa taajuusalueen haluamallaan tavalla kanaviin, mutta EU433 ja EU868 luokitukset määrittävät kolme pakollista kanavaa 125kHz taajuuksilla taulukoiden 2.2 ja 2.3 mukaisesti [26]. Pakollisia kanavia hyödynnetään esimerkiksi päätelaitteen liittyessä ensimmäistä kertaa verkkoon. Vastaavat pakolliset kanavat löytyvät jokaisesta maakohtaisesta luokituksesta käytössä olevan taajuusalueen mukaisesti. Globaalisti muita mahdollisia kanavataajuuksia ovat 250kHz ja 500kHz. Näistä Euroopassa käytetään pääsääntöisesti 125kHz ja 250kHz taajuuksia [39].

Taulukko 2.2: EU868 oletuskanavat [26]

Modulaatio	Kaista kHz	Kanvataajuus	Modulointi LoRa tai FSK Siirtonopeus	Kanavien määrä	Käyttösuhde
LoRa	125	868.10 868.30 868.50	DR0 - DR5 0.3 - 5kbps	3	<1 %

2.2.2 Verkonrakenne ja verkkopalvelu

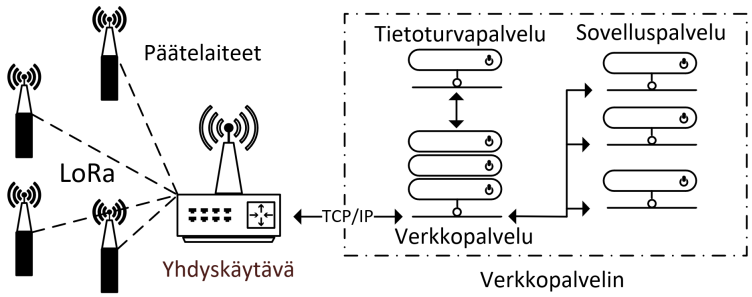
LoRaWAN-verkkotopologia kuvataan artikkelissa [6][25] seuraavasti. *LoRaWAN* verkko on rakenteeltaan tähtimäinen ja jokainen päätelaite on kytkeytynyt yhdyskäytävän kautta verkkoon. Päätelaite voi olla yhteydessä yhteen tai useampaan yhdyskäytävään samanaikaisesti. Yhdyskäytävät kytkeytyvät verkkopalvelimelle standardilla IP-yhteydellä. Päätelaiteviestintä yhdyskäytävälle perustuu LoRa-modulaatioon

Taulukko 2.3: EU433 oletuskanavat [26]

Modulaatio	Kaista kHz	Kanvataajuus	Modulointi LoRa tai FSK Siirtonopeus	Kanavien määrä	Käyttösuhde
LoRa	125	433.175 433.375 433.575	DR0 - DR5 0.3 - 5kbps	3	<1 %

käytettäessä yhtä yhdyskäytävää.

LoRaWAN-verkko sisältää verkkopalvelimen, jonka roolin voi jakaa kolmeen palveluosaan. Verkkopalvelin voi olla yksi tai useampi fyysinen tai virtuaalinen laite, joihin palveluroolit sijoittuvat (kts. 2.4). Verkkopalvelu hallitsee verkon MAC-osoitteita, tietovuota kuten kehyslaskentaa ja kuittauksia. Verkkopalvelu toimii myös yhdyskäytävänä tietoturva- ja sovelluspalvelun välillä. Tietoturvapalvelu vastaa päätelaitteiden ja sovellusten autentikoinnista, auktorisoinnista ja autentikointiavainten jakelusta ja hallinnasta. Sovelluspalvelu toimii yhdyskäytävänä verkkopalvelun ja varsinaisen sovelluksen välillä. Sovelluspalvelun avulla voidaan myös kontrolloida päätelaitteiden L2-luokituksia (kuva 2.3), käytettäviä kanavataajuuksia, ADR-arvoja ja muita tietoliikennettä koskevia määreitä. ADR-käsitettä tarkastellaan tarkemmin luvussa 2.2.5



Kuva 2.4: *LoRaWan* verkkopalvelu [6]

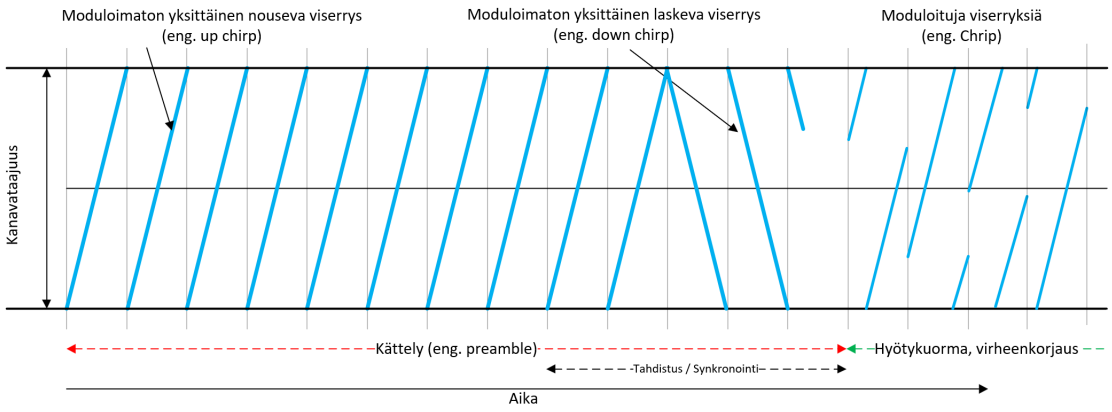
LoRaWAN-spesifikaatio määrittää yhden hypyn (engl. single-hop) päätelaitteen ja yhdyskäytävän välillä, eli päätelaitteet eivät uudelleen välitä saapuneita paketteja [27]. Näin ollen voidaan todeta, että *LoRaWan* verkon peittoalue määräytyy sen yhdyskäytävien kuuluvuuden mukaisesti ja kuuluvuuteen vaikuttavat ympäristötekijät kuten esteettömyys, häiriöttömyys ja laitteiden tietoliikenne ominaisuudet.

2.2.3 Kanavallepääsy

LoRaWAN käyttää kanavalle pääsyssä ALOHA-protokollaa, joka skaalautuu huonosti mikäli verkko sisältää suuren määrään päätelaitteita [27]. ALOHA-protokollasta hyödynnetään sen yksinkertaisinta versiota *LBT* (engl. Listen-Before-Talk), missä kanavalle pääsy perustuu pelkästään kanavan vapaana olemisen kuunteluun [21]. Toiminta soveltuu hyvin A-luokan päätelaitteille, mutta B-luokan päätelaitteiden osalta tämä tarkoittaa monimutkaisempaa aikaikkunahallintaa verkkopalvelimella törmäyksien välttämiseksi [21].

2.2.4 LoRaWAN kehyksen kättely vastaanottajalla

Kuvassa 2.5 on esitetty kättelyn (engl. preamble) toimintaperiaate. Kehyskättelyn avulla varmistetaan, että vastaanottaja tunnistaa *SF* hajautuskertoimen ja synkronoituu oikealle aikajaolle [27]. Kättely perustuu euroopassa yhteensä 12,25 moduloimattomaan viserrykseen, jotka voidaan esittää kuvan 2.5 mukaisella tavalla [26]. Kättely alkaa 10 nousevalla viserryksellä (engl. up chirp) ja lopussa on 2,25 laskevaa viserrystä (engl. down chirp). Viimeistä 4,25 viserrystä käytetään oikean aikaikkunan synkronointiin vastaanottajalla. Kättelyn jälkeen seuraa varsinainen hyötykuorma [45]. Modulaatiota ja hajautuskertoimen *SF* toimintaa kuvataan tarkemmin luvussa 2.1.



Kuva 2.5: *LoRaWAN* kehyskättely

2.2.5 ADR

ADR (engl. Adaptive Data Rate) on *LoRaWAN* ominaisuus, jonka avulla voidaan muuttaa päätelaitteen *DR* (engl. data rate) arvoja määrittämällä vaihoehtoisia *SF*, *BW* tai lähetysteho arvoja. Tarkoituksena on vähentää virrankulutusta, optimoida lähetysaikoja ja käyttää päätelaitteen kannalta edullisinta yhdyskäytävää liikennöinnissä. Verkkopalvelu kerää tiedonsiirtoon liittyvää tietoja ja voi tämän perustella pyytää päätelaitetta muutamaan asetuksiaan. Jokainen päätelaite päättää itsenäisesti soveltaako se verkkopalvelun tarjoamia ARD-arvoja [6].

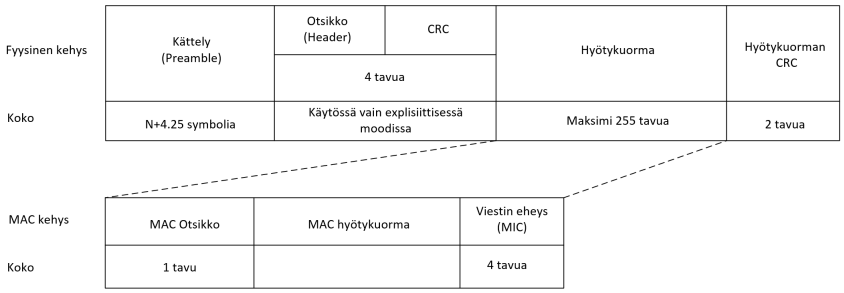
2.2.6 Tietoturva

Kaikessa tietoliikenteessä tulee kiinnittää huomiota tietoturvaan. Tietoturvan toteuttamiseen liittyvät prosessit eivät saa kuitenkaan haitata verkon skaalautuvuuteen, käytettävyyteen, hallintaan tai virrankulutukseen liittyvää toimintaa [13]. *LoRaWAN* protokolla mahdollistaa salauksen niin päätelaite kuin sovellustasolla standardilla AES- algoritmilla [20]. Jokaisella päätelaitteella ja sovelluksella on oma yksilöllinen tunnisteensa, jota hyödynnetään salausavaimien luonnissa ja hallinnassa [20]. Päätelaitteiden ja sovellusten salausavaimet ovat sessioperusteisia ja niitä hallinnoidaan verkkopalvelimen toimesta [20]. *LoRaWAN* päätelaite pystyy liittymään verkkoon kahdella eri tavalla. Todentaminen voidaan suorittaa ABP-prosessin kautta, jolloin päätelaitteesta toimitetaan kaikki tarvittava tieto verkkopalvelimelle, joka muodostaa tarvittavat avaimet liikennöintiä varten. ABP-todentamista varten tarvitaan operaattori (ihminen) suorittamaan prosessin vaiheet [6]. Päätelaite voi myös pyytää itsenäisesti todennusta OTAA-prosessin kautta. Prosessi käynnistyy päätelaitteen kättelyllä verkkopalvelulle ja tämä tapahtuu aina kun uusi laite liittyy verkkoon tai se muodostaa uudelleen katkennutta yhteyttä. Mikäli verkkopalvelu tunnistaa päätelaitteen toimittaa se tälle sessioavaimen liikennöintiä varten. [6]

2.2.7 Kehyksen rakenne

LoRaWAN-kehyksen toimintaperiaate kuvataan julkaisussa [32] seuraavasti. Kehys muodostuu kättelystä (engl. preamble), jota käytetään jokaisella kehyksellä liikenteen alustukseen ja aikatahdistukseen (kuva 2.6). *LoRaWAN* hyödyntää explisiistä ja implisiittistä kehysmuotoa liikennöinnissä. Implisiittisessä kehyksessä ei ole lainkaan otsikko- ja CRC-tietoa kts. kuva 2.6 Käytettäessä explisiittistä muotoa ilmais-

taan kehyksessä *FEC* (engl. Forward error correction) arvo, hyötykuorman koko ja tieto *CRC* (engl. Cyclic redundancy check) arvon mukana olosta, joka on valinnainen. Hyötykuorman *CRC*- arvolla voidaan tarkistaa hyötykuorman eheys. Hyötykuorma jakautuu *MAC* (engl. medium access controll) otsikkoon, jossa on tieto hyötykuorman sisällöstä. Tiedossa voidaan kuvata onko kyseessä dataa tai esimerkiksi tilatietoa. *MIC* (engl. Message Integrity Code) bittien avulla varmistetaan tiedon muuttumattomuus ja tiedon alkuperän aitous. Tätä kehyksen osaa käytetään esimerkiksi kun päätelaite on liittymässä osaksi verkkoa.



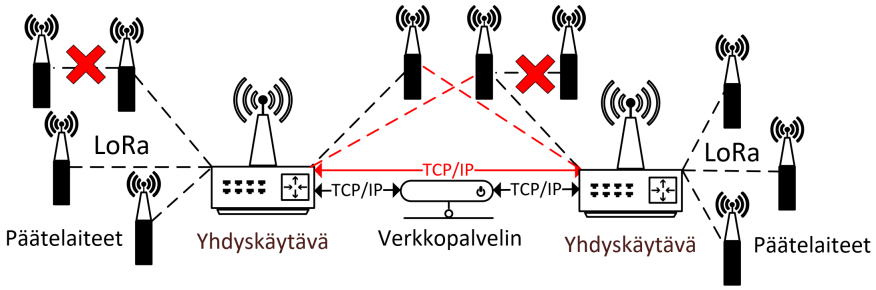
Kuva 2.6: *LoRaWAN*-kehys [32]

3 Mesh toteutukset LoRa-moduloinnilla

Tässä luvussa ja sen alaluvuissa tarkastellaan, miten *LoRa*-modulointiin perustuvan verkon peittoaluetta voidaan laajentaa. Kysymys on monitahoinen ja tarkastelukulmia ratkaisutapoineen on monia. Kompleksisuus muodostuu osaltaan siitä, että kohteena voi olla kaksi protokollaa. *LoRaWAN* ei voi toimia ilman *LoRa*-protokollan tuottamia tiedonsiirron palveluita. Vastaavasti *LoRa* ei sellaisenaan tarjoa hyötykuorman, tietoturvan tai esimerkiksi datavuon käsittelyyn liittyviä ohjelmallisia palveluita vaan kehittäjän on käytettävä jotain valmista sovelluskirjastoa tai muodostettava nämä itsenäisesti. Esimerkki tällaisesta *LoRa* kirjastosta on Radiohed [29]. Tarkastelu perustuu ainoastaan *LoRa*-moduloituun tiedonsiirtoon, jossa voi olla mukana *LoRaWAN* verkkopalveluita.

Yksinkertaisin tapa kasvattaa *LoRaWAN*-verkon kuuluvuusaluetta on lisätä yhdyskäytävien määrää. Toteutustapa kasvattaa kustannuksia ja saattaa joissain tapauksissa olla mahdoton toteuttaa ympäristörajoitteiden kuten laitteiden fyysisen sijainnin takia. Yhdyskäytävän fyysinen sijainti voi vaikuttaa jatkuvan sähkön saatavuuteen tai TCP/IP-yhteyteen, jonka avulla yhdyskäytävä on yhteydessä verkkopalveluun. *LoRaWAN*-verkossa olevat päätelaitteet voivat olla yhteydessä useampaan yhdyskäytävään kerrallaan, ja jokaisella yhdyskäytävällä pitää olla TCP/IP yhteys *LoRaWAN*-verkkopalvelimelle. Tilanteessa, missä dataa haluaan välittää päätelaitteelle tapahtuu se yhden yhdyskäytävän toimesta kerrallaan. *LoRaWAN*-verkko voidaan kuvata kuvan 3.1 tavalla. Kuvassa punaisella kuvatut linkit ovat vaihtoehtoisia ja päätelaitteet eivät voi välittää dataa toistensa välillä. Ilman erillistä reititystä, kahden *LoRaWAN*-yhdyskäytävän välinen TCP/IP yhteys on oltava samassa IP-aliverkossa.

Mesh-verkko (engl. mesh network) on verkkotyyppi, joka koostuu useista yksittäisistä laitteista, joita kutsutaan solmuiksi (engl. nodes). Solmut ovat yhteydessä toisiinsa muodostaen itsenäisen ja hajautetun verkon, joka kytkeytyvät toisiinsa ilman yksittäistä keskitettyä verkkopalvelinta tai yhdyskäytävää. Mesh-verkot ovat itseorganisoituvia ja joustavia, mikä tarkoittaa sitä, että solmut voivat kommunikoida keskenään välittämällä tai reitittämällä tietoa toisilleen. Tieto voi kulkea monia eri reittejä pitkin, mikä tekee mesh-verkoista skaalautuvia ja vikasietoisia. Jos yksi



Kuva 3.1: *LoRaWAN*-periaate

solmu epäonnistuu, mesh-verkko voi löytää uusia reittejä datan siirtämiseksi kohteeseen. [17] Esimerkki tällaisesta mesh-verkosta kuvataan luvussa 4.3.

Verkko voi toimia myös osittain mesh-periaatteella, jolloin verkossa oleva yksittäinen laite hallitsee esimerkiksi aikaikkunoinnin tai reititystiedon muodostusta ja kontrollointia. Edellä mainittu laite toimii verkon puheenjohtajana ja verkko toimii muuten aiemmin kuvattujen Mesh periaatteiden mukaisesti. Tällainen toiminta on ominaista IoT-Mesh verkoissa, joissa käytössä on jokin reititysprotokolla ja IoT-päätelaitteet välittävät tai reitittävät tietoa keskenään.

IoT Mesh-verkojen toiminta soveltuu hyvin käyttötapauksiin, missä internet ei ole käytettävissä tai sen käyttö ei ole kustannustehokasta. IoT Mesh-verkot soveltuvat myös hyvin käyttötapauksiin, missä verkko pitää muodostua automaattisesti ja sen topologian ylläpidosta huolehtii jokin verkossa oleva laite. Mesh-verkoissa voidaan hyödyntää reitittäviä ja välittäviä ratkaisuja.

3.1 Välittävät ratkaisut

Datan välittäminen (engl. data forwarding) tarkoittaa laitteen prosessia, jossa se vastaanottaa datan ja lähettää sen eteenpäin kohti sen lopullista määränpäätä. Dataa välittävä laite ei välttämättä tunne tai tiedä datan lopullista määränpäätä tai vastaanottavaa laitetta. Välittäminen on tärkeä osa verkon toimintaa, koska se mahdollistaa datan siirtämisen eri verkkolaitteiden välillä ja sen saavuttamisen oikeaan lopulliseen kohteeseen.

Monihyppy (engl. multihop) tarkoittaa viestienvälitystä useiden eri verkkolaitteiden välillä, jotta viesti saavuttavat määränpäänsä. Monihyppy on hyödyllinen ominaisuus langattomissa verkoissa, koska se mahdollistaa verkon laajentamisen

alueille, jotka muuten saattaisivat olla esimerkiksi yhdyskäytävän tavoittamattomissa [23]. Monihyppyperusteinen viestinvälitys mahdollistaa verkon toiminnan myös häiriötilanteessa, sillä viestit voivat löytää uusia reittejä toisten laitteiden kautta ja taata tällä tavoin verkon toiminnan. Monihyppyverkoissa laite voi toimia sekä lähettäjänä että vastaanottajana, ja viestit kulkevat verkon läpi useiden eri vaihtoehtoisten reittien kautta. Uuden laitteen lisääminen monihyppyverkkoon on yksinkertaista, koska verkon viestinvälitys mukautuu saatavilla olevien laitteiden mukaisesti. Koska tiedonvälityksessä voi syntyä useita reittejä, tulee monihyppyverkoissa kiinnittää erityistä huomiota silmukoiden estoon (engl. loop prevention). Silmukalla tarkoitetaan tilannetta, missä aiemmin lähetetty data palaa lähettäjälle uudelleen toista reittiä pitkin. Silmukoiden syntymistä voidaan estää käyttämällä välitys- tai reititysalgoritmeja tai tarkastelemalla pakettien uudelleen ilmenemistä samalla laitteella.

3.1.1 Välittävät päätelaitteet

Julkaisussa [3] esitellään LoRaBlink protokolla, joka ei ole *LoRaWAN* yhteensopiva. LoRaBlink mukailee TDMA (engl. Time Division Multiple Acces) toimintaperiaatteita. TDMA-protokolla määrittää yksiselitteisesti, milloin kukin laite saa käyttää verkkoa lähettämiseen tai vastaanottamiseen. TDMA jakaa käytettävän ajan yhtä pitkiin aikajaksoihin (engl. time slots). Jokainen laite varaa itselleen yhden tai useamman aikajakson, jonka aikana se voi lähettää tai vastaanottaa tietoja. Kun aikaväli on varattu laitteelle, se voi käyttää taajuutta vapaasti aikaikkunan puitteissa.

LoRaBlink toimii monihyppy periaatteella ja on tarkoitettu vähävirtaiseksi, vikasietoiseksi ja kaksisuuntaiseksi matalan viiveen IoT-protokollaksi. Ehdotuksen toimintalogiikka perustuu ennalta määriteltyjen aikaikkunoiden käyttämiseen datan lähettämisessä ja vastaanottamisessa. Verkonmuodostuksen aikana dataa lähetetään yhdyskäytävältä tulvimalla (engl. multicast) niin, että kaikki kuuluvuusalueella olevat päätelaitteet vastaanottavat sen ja vastaavat siihen. Tilatessa missä päätelaite haluaa lähettää dataa yhdyskäytävälle tai toiselle päätelaitteelle, lähetetään se täsmälähetyksenä (engl. unicast). Toteutuksen aikaikkunointia hallitaan yhdyskäytävän lähettämillä majakka (engl. beacon) viesteillä. Kun päätelaite vastaanottaa yhdyskäytävän majakkaviestin, se tulvii oman majakkaviestinsä eteenpäin muille mahdollisesti pidemmällä oleville päätelaitteille. Verkon puumaisen topologian luominen ja aikasynkronointi tapahtuu majakkaviestien lähtemisen ja vastaanottamisen avulla. Tilanteessa missä päätelaite haluaa lähettää dataa, lähettää se aina sen

sille laitteelle, jolta se on vastaanottanut majakkaviestinsä. Vastaanottanut päätelaite uudelleen välittää saamansa datan yhdyskäytävälle tai vastaavasti sille päätelaitteella, joka on lähempänä yhdyskäytävää majakkaviestin perusteella.

Kanavan vapaanaoloa, kuten myös tarkastelua siitä pitääkö päätelaitteen olla hereillä vastaanottoa varten, hallinnoidaan aikaikkunoihin sidottujen CAD-syklien (engl. carrier activity dedection) avulla. Mikäli kanavalta kuullaan kättelyyn (eng. preabmle) liittyviä taajuusmuutoksia herätetään radio täysin vastaanottoa varten. Toteutuksella käytetty Semtech SX1272 vastaanotin tukee kättelyyn liittyvää kanavankuuntelua ilman, että radio on täysin päällä ja tämän avulla kyetään säästämään laitteen virtaa. LoraBlink hyödyntää LoRa:n ominaisuutta, missä yhtäaikaisen eri hyötykuormasisällön lähettäminen ja vastaanottaminen samalla taajuudella ja hajautuskertoimella on mahdollista. Julkaisussa on tarkasteltu yhtäaikaisten lähetyksien toimivuutta ja tämän osalta voidaan todeta, että tekniikka toimii verrattain luotettavasti, kun laitemäärät ovat pieniä.

3.1.2 Välittävät yhdyskäytävät

Yhdyskäytävä voi olla fyysinen laite tai se voi toimia osana reititintä laajentaen sen tuottamia palveluita. Yhdyskäytävä voi myös toimia osana kytkintä tai PC- tai IoT-laitetta. Yhdyskäytävä kykenee tekemään protokolla ja mediamuunnoksia, kuten esimerkiksi *LoRaWAN*-pakettien muuntamisen TCP/IP-paketeiksi eli sille on ominaista yhdistää protokolla- tai mediatyypiltään poikkeavia verkkoja toisiinsa. Esimerkiksi mediamuunnos langattomasta mediasta langalliseen mediaan voi tapahtua yhdyskäytävän toimesta.

Seuraavassa esitellään toteutus, joka on julkaisijoidensa mukaan päätelaiteperusteinen, mutta kuitenkin mukailee yhdyskäytävälle ominaisia toimintatapoja.

Julkaisulla [40] esitellään toimintatapa ja tekniikka, jonka avulla voidaan laajentaa *LoRaWAN*-verkon peittoaluetta mahdollistamalla päätelaitteen tiedonvälitys. Julkaisijoiden mukaan kyseessä on tekniikka, joka on yhteensopiva olemassa olevan *LoRaWAN*-teknologian kanssa, muuttamatta standardin määrittämää toimintatapaa. Käytännössä tämä tarkoittaa *LoRaWAN*-verkkoon sijoittuvaa päätelaitetta, joka käyttäytyy kuten A- tai B-luokituksen päätelaite ja kykenee myös tarvittaessa uudelleen välittämään *LoRaWAN*-dataliikennettä.

LoRaWAN-verkossa voidaan käyttää eri hajautuskertoimia ja taajuusalueita yhtäaikaisesti. Useamman hajautuskertoimen ja taajuusalueen yhtäaikainen käsitteleminen edellyttää monikanavaisen Lora-radion käyttämistä. Julkaisussa olevassa

päätelaitteessa on käytetty Laird RG186 mikrokontrolleriin ja SX1301 LoRa-radion yhdistelmää. SX1301-radiota käytetään yleisesti C-luokan yhdyskäytävän radiona. Uudeksi päätelaiteluokaksi ehdotetaan E-luokkaa ja toimintaperiaatteesta löytyy yhtäläisyyksiä C-luokan kanssa. Kun päätelaite toimii E-luokan tilassa, käyttäytyy se kuten C-luokan yhdyskäytävä välittäen dataa tarkoitusta varten muokatulle verkkopalvelulle. Päätelaite välittää saamiaan paketteja UDP/IP perusteisesti JSON-formaatissa ethernet-yhteyden avulla. Muokattu verkkopalvelu on toteutettu IBM:n kehittämällä Node-RED sovelluskehittimellä. Node-RED verkkopalvelu välittää käsittelemänsä datan *LoRaWAN*-verkkopalvelulle. Ehdotettu toimintalogiikka perustuu tiedon kaksinkertaiseen lähettämiseen eri aikaikkunoissa. Tämän avulla varmistetaan, että lähetty data on saavuttanut *LoRaWAN*-verkkopalvelun.

Soveltuvuusselvityksen yhteydessä toteutusta on tarkasteltu lähettämällä dataa kahden ennalta määriteltyyn aikaikkunan puitteissa verkkopalvelulle. Tekniikka soveltuu käytettäväksi ainoastaan alhaisilla 7-9 hajautuskertoimilla johtuen aikakatkaisusta, jonka avulla tarkastellaan, onko tieto saavuttanut yhdyskäytävän määritellyn aikaikkunan puitteissa.

Julkaisussa [41] esitellään käytännöntoteutus perustuen edellä esiteltyyn E- luokituksen perustuvaan tekniikkaan. Toteutuksessa E-luokan päätelaite tuottaa *LoRAWAN*-yhteensopivan, läpinäkyvän yhden hypyn verkon. E-luokan päätelaitteita ei voi olla ketjussa toisiinsa nähden. Ehdotuksen verkko muodostu A- tai B-luokan vakiopäätelaitteesta ja mahdollisesta E-luokan välittävästä päätelaitteesta. Verkossa käytetään vakio *LoRaWAN*-yhdyskäytävää. Läpinäkyvyys tarkoittaa, että E-luokan päätelaite välittää viestin ainoastaan siinä tapauksessa, että yhdyskäytävä ei vastaa tietyn aikakatkaisun puitteissa alkuperäiseen A- tai B-luokan päätelaitteen viestiin. Tämä mahdollistaa samalla myös sen, että verkkopalvelu näkee ja käsittelee viestin vain kerran. Ehdotuksen prototyyppitoteutus nojaa kiinteän virtalähteen omaaviin Rasberry Pi laitteisiin ja niissä oleviin Laird RG186-M2 radiomoduuleihin, jonka avulla prototyyppi kykenee käsittelemän useamman hajautuskertoimen dataa yhtäaikaisesti. Rasperry PI laitteita on kaksi, joista toinen vastaa datan lähetyksestä ja toinen vastaanotosta. Data välitetään prototyyppi päätelaitteeella JSON-formaatissa Node-RED sovellukselle, missä varsinainen datan käsittely tehdään. JSON-tiedonsiirrossa käytetään Semtechin kehittämää välitystekniikkaa (engl. basic packet forwarder) [36], jonka avulla LoRa-kehyksen sisältö voidaan välittää UDP/IP muodossa. Node-RED purkaa saapuneen datan, tarkistaa paketin aikaleimatiedon mistä päätellään, onko uudelleen lähetystä tarpeellista tehdä. Samas-

sa yhteydessä paketti puskuroidaan mahdollista uudelleen lähetystä varten. Samaa logiikkaa noudatetaan tilanteessa, missä dataa ollaan välittämässä kohti päätelaitetta. Ehdotuksen testaustuloksien kannata voidaan todeta, että logiikka toimii ja verkon peittoaluetta voidaan kasvattaa kun käytössä on hajautuskertoimet 7-9.

3.1.3 Yhteenveto välittävistä ratkaisuista

Välittäville ratkaisuille on ominaista, että lähettävän laitteen ei tarvitse tietää kuinka kaukana se on yhdyskäytävästä tai ylipäätään kohdelaitteesta. Välittämisessä käytetty protokolla vastaa siitä, että tietoa välitetään kaikille tarpeellisille osapuolille ja välittämiseen voi liittyä useita uudelleen välityksiä eli monihyppyjä. Yksinkertaisimmillaan tiedon välittäminen on moninkertaista tulvimista (engl. multicast), jolloin jokainen saapunut paketti uudelleen välitetään sellaisenaan niin kauan, että se saavuttaa määränpäänsä. Esimerkki tällaisesta verkosta esitellään pro gradun käytännön sovellus osuudessa. Välityspäätökset voivat myös perustua aiemmin esiteltyyn majakkaviestien määrittämään rakenteeseen, jolloin tulvimisen sijaan viestiä välitetään laitteelta laitteelle täsmälähetyksenä (engl. unicast). Tässäkin tapauksessa on kuitenkin huomioitava, että verkontopologiaa ylläpidetään tulvimalla majakkaviestiä kaikkien siellä olevien laitteiden toimesta.

Tulvimiseen perustuvassa toteutuksissa ei tiedetä, mitä reittiä pitkin dataa kuljetetaan, joka tekee verkon hallinnasta autonomista. Ominaisuus tekee verkon toiminnasta vikasietoista niin laiterikkojen kuin tietoliikenteeseen liittyvien häiriöiden osata. Kaikissa tulvimiseen perustuvissa ratkaisuissa laitemäärät näyttelevät merkittävää roolia verkon kuormituksen kannalta. Jos samalla taajuudella ja hajautuskertoimella varustettuja laitteita on suuri määrä saman kuuluvuusalueen sisäpuolella voi verkko alkaa ylikuormittumaan tulvimisen takia. Ylikuormittuminen esiintyy verkon datavälityksen hidastumisena tai toimimattomuutena.

Riippumatta totutustavasta verkossa oleva sovellus tai verkkopalvelu voi saada saman datan useita eri reittejä pitkin eri aikajaksoissa. Sovelluksen tai verkkopalvelun on kyettävät toimimaan normaalisti huolimatta tästä ominaisuudesta. Välittäminen kuten myös vastaanottaminen edellyttää laitteen valveillaoloa, joten virrankulutuksen kannalta kyseessä ei ole energiatehokas toimintamalli. Virrankulutusta voidaan hallita aikaikkunoiden avulla, mutta samalla se nojaa liikennöinnissään enemmän tiedossa olevaan topologiaan, joka välttämättä ei ole ajantasainen varsinkaan jos sitä ei ylläpidetä aktiivisesti.

Taulukko 3.1: Välittävät ratkaisut

Julkaisu	Tyyppi	Toteutustapa
Aslam et al. [2]	Välittävä-yhdyskäytävä Reitittävä-yhdyskäytävä	LoRaWAN
Ebi et al. [11]	Reitittävä-päätelaite Välittävä-päätelaite	LoRa
Choi et al. [5]	Välittävä-päätelaite	LoRa
Mamour et al. [31]	Välittävä-päätelaite	LoRa
Tanjung et al. [43]	Välittävä-päätelaite	LoRaWAN
Duong et al. [9]	Välittävä-päätelaite	LoRa
Abrardo et al. [1]	Välittävä-päätelaite	LoRa
K. Hester et al [19]	Välittävä-päätelaite	LoRa
Bor et al. [3]	Välittävä-yhdyskäytävä Välittävä-päätelaite	LoRa
Liao et al. [22]	Välittävä-päätelaite	LoRa
Mai et al. [30]	Välittävä-päätelaite	LoRa
Zhu et al. [47]	Välittävä-päätelaite	LoRaWAN
Sisinni et al.[40][41]	Välittävä-yhdyskäytävä	LoRaWAN
Zhou et al. [46]	Välittävä-päätelaite	LoRaWAN

Taulukossa 3.1 listataan muita välittäviä ratkaisuehdotuksia niin LoRa kuin LoRaWAN perusteisina toteutuksina.

3.2 Reitittävät ratkaisut

Reitittäminen (engl. routing) tarkoittaa toimintaa, jossa verkkolaitteet valitsevat optimaalisen polun datan siirtämiseksi lähettäjältä vastaanottajalle. Reititetty verkko voi olla rakenteeltaan langallinen tai langaton. Reitittäminen on keskeinen osa verkon toimintaa, ja se mahdollistaa datan liikkumisen eri verkkojen välillä. Reitittämisellä viitataan kahden eri verkon väliseen liikennöintiin.

Kun verkkolaite lähettää dataa, lähettää se sen siihen verkkoon mihin se on kytketty. Verkko voi koostua useista eri laitteista, kuten reitittimistä, yhdyskäytävästä ja kytkimistä ja muista päätelaitteista. Verkossa oleva reititin päättää tai päättelee,

mikä on paras reitti datan siirtämiseksi vastaanottajalle reititysprotokollan avulla. Reitityspäätös perustuu erilaisiin kriteereihin, kuten verkon ruuhkautumiseen, nopeuteen, matkaan eli hyppyjen määrään ja kustannuksiin.

Reitityspäätös tapahtuu pääsääntöisesti reititystaulukoiden avulla, jotka sisältävät tietoa verkossa olevista laitteista ja niiden yhteyksistä muihin reitittäviin laitteisiin. Tämän tiedon perusteella reititin päättää, mihin suuntaan data tulee lähettää seuraavaksi. Reititystiedon perusteella reititin lähettää datan seuraavaan verkkolaitteeseen, joka päättää, minne se seuraavaksi siirretään. Reitittäminen mahdollistaa datan siirtämisen eri verkkojen läpi. Se auttaa myös estämään verkon ruuhkautumista ja varmistamaan, että data saavuttaa oikean vastaanottajan mahdollisimman nopeasti.

Puhuttaessa reitittimestä viitataan sillä laitteessa olevaan ohjelmistoon tai reititinlaitteeseen, joka kykenee vastaanottamaan, analysoimaan ja reittittämään verkossa liikkuvia paketteja. Reititin reitittää datapaketteja reititysprotokollan määrityksen mukaisesti vaikka verkon mediatyyppi vaihtuisi reitityksen aikana. Reititys muodostaa rajan reititettävien verkkojen välille ja liikennöinti näiden välillä on mahdollista ainoastaan reitityksen avulla. Tyypillisesti laajemmissa IoT-verkoissa olevat päätelaitteet kykenevät reitittämään dataa. Tätäkin toimintamallia on esitelty pro gradun käytännön osuudessa.

3.2.1 Reitittävät päätelaiteratkaisut

DSDV (Destination-Sequenced Distance Vector) on langattomien ad hoc -verkkojen reititysprotokolla ja sen toiminta perustuu etäisyysvektoriin laskentaan eli verkossa tapahtuvien hyppyjen lukumäärään. DSDV:n keskeinen ominaisuus on, että jokaisella verkon päätelaitteella on oma reititystaulukkonsa, joka sisältää tietoa verkon muiden solmujen kautta saavutettavista solmuista ja niiden välisistä etäisyyksistä. Jokainen päätelaite päivittää käyttämäänsä reititystaulukkoa itsenäisesti, ja jokainen solmu lähettää tietoa naapurisolmuilleen verkon tilasta.

Julkaisussa [8] esitellään yksinkertaistettu versio DSDV-reititysprotokollasta, jossa langattoman verkon reitystä toteuttaa muokattu päätelaitetyyppi. Ehdotuksessa liikennöinti toimii ainoastaan reunapäätelaitteelta kohti yhdyskäytävää ja se on *LoRaWAN*-yhteensopiva yhdyskäytävän osalta. Reitittävät-päätelaitteet (myöhemmin ReP) ja reunapäätelaitteet, jotka tässä kontekstissa voidaan mieltää A- tai B-luokan päätelaitteiksi eivät noudata *LoRaWAN*-yhteensopivuutta johtuen julkaisussa esiteltävästä reititysprotokollasta. Reititys olettaa, että ReP-laitteiden sijainti ei muutu ja

ne ovat jatkuvasti saavutettavissa. Toimintalogiikka perustuu yhdyskäytävän lähettämiin *LoRaWAN*-majakka (eng. beacon) viesteihin, joiden avulla ReP-laitteella oleva DSDV-protokolla muodostaa reititystiedon. ReP-laite lähettää oman reititystietonsa broadcast viestinä reunapäätelaitteille. Tilanteessa, missä reunapäätelaite haluaa lähettää dataa yhdyskäytävälle, muodostaa se datapaketin kapseloimalla *LoRaWAN*-datan osaksi reititysprotokollan pakettia ja välittää tämän reitityksen määrittämälle ReP-laitteelle. Riippuen verkon topologiasta Rep-laite välittää datapaketin joko seuraavalle Rep-laitteelle tai suoraan yhdyskäytävälle. Yhdyskäytävää lähinnä oleva Rep-laite välittää ainoastaan LoRaWAN-dataa sisältävän paketin ja tällä mahdollistetaan *LoRaWAN* yhteensopivuus yhdyskäytävän osalta.

Ehdotettu verkko käyttää ReP- ja reunapäätelaitteen osalta taajuusväliä 868.0 MHz-868.6 MHz. Rep- tai reunapäätelaitteelle käytetään liikennöinnissään satunnaisesti kanavataajuuksia 868.1, 868.3 tai 868.5 MHz. Yhdyskäytävä lähettää majakkaviestit ReP-laitteelle kanavataajuudella 869.525 Mhz. Rep- ja reunapäätelaitteet käyttävät eri hajautuskerrointa kuin yhdyskäytävä majakkaviesteissä.

Kanavan vapaanaolon tarkistaminen on toteutettu CAD (engl. carrier activity detection) kaltaisella mekanismilla, missä seurataan tiedonsiirron kättelyä (engl. preamble). Mikäli kättelyyn liittyvää viestintää ei kuulla voidaan aloittaa lähetys. Mikäli kättely kuullaan odotetaan vähintään se aika, minkä kyseisen lähetyksen lähettäminen olisi varannut verkolta. Maksimi odotusaika on edellä mainittu aika lisättynä satunnaisella odotusajalla.

Julkaisussa esitellyssä toteutuksessa ReP-päätelaitteina toimii STM32L432KC mikroohjaimeen perustuva laite, ja reunapäätelaitteet perustuvat Arduino tuoteperheen mikro-ohjaimiin. Molemmat päätelaitetyypit käyttävä HopeRF:n valmistamaa RFM95-radiomoduulia. Yhdyskäytävänä joka tuottaa verkko- ja sovelluspalvelut toimii Rasperry Pi johon on lisätty iC880A-radiomoduuli.

Tekniikan toimivuutta on tarkasteltu PRR (engl. Packet reception rate) avulla. PRR, eli pakettien vastaanottoprosenttia käytetään, kun halutaan tarkastella verkon toiminnan luotettavuutta. Se on mittari, joka ilmaisee, kuinka monta lähetettyä pakettia on vastaanotettu onnistuneesti. Tätä mittaria käytetään arvioimaan verkon suorituskykyä ja luotettavuutta, erityisesti langattomissa verkoissa, joissa tiedon lähettäminen ja vastaanottaminen voivat olla haastavaa häiriöiden tai signaalin heikkenemisen vuoksi. PRR:n laskentakaava on yksinkertainen: PRR = (vastaanotetut paketit / lähetetyt paketit) * 100%. Jos esimerkiksi lähetetään 100 pakettia ja vastaanotetaan 90 niistä, PRR on 90%.

Julkaisun mittautuloksissa on päästy yli 90% luotettavuuteen tiedonvälityksessä.

3.2.2 Reitittävät yhdyskäytäväratkaisut

Julkaisussa [28] esitellään reititysprotokolla, jossa reitittäminen perustuu HWMP (engl. hybrid Wireless Mesh Protocol) ja AODV (engl. Ad-hoc On Demand Distance Vector Routing) protokollien yhdistelmään. Reititys tapahtuu *LoRa*-tasolla ja on julkaisijoidensa mukaan *LoRaWAN*-yhteensopiva, sillä edellytyksellä, että käytössä on julkaisussa esitelty reititysprotolla verkossa olevien yhdyskäytävien välillä. Ehdotuksessa AODV-protokollasta on tarkoitusta varten mukautettu toimintaperiaate, minkä avulla yhdyskäytävä pyytää itselleen reititystietoa tulvimalla kuuluvuusalueelleen RREQ (engl. route request) paketin. Jokainen paketin vastaanottanut yhdyskäytävä tallentaa lähdetiedon itselleen ja rakentaa tämän perusteella reititystietoa. Useampi yhdyskäytävä voi vastata pyyntöön RREP (engl. route reply) paketilla, jonka perusteella alkuperäinen pyytäjä muodostaa reititystietoa. Mikäli alkuperäistä pyytäjää ei tavoiteta, tulvitaan REPP (engl. route error) paketti, jotta kaikki RREQ-paketin vastaanottaneet tietävät yhdyskäytävän olevan poissa kuuluvuusalueelta.

HWMP-protokolla on tarkoitettu puumaisen reititystopologian muodostamiseen. Reititys voi toimia niin, että verkossa on yksi juurilaite, joka vastaa verkon reititystiedon ylläpitämisestä. Tämä piste toimii samalla yhdyskäytävänä *LoRaWAN*-verkon ja internetin välillä.

Ehdotuksen reititysprotokolla toimii tunnelointiperiaatteella ja on muokattu versio HWMP- ja AODV-protokollista. Kun yhdyskäytävä saa itselleen paketin päätelaitteelta ja se ei omaa internetyhteyttä, reitittää se paketin edelleen reititystietonsa perusteella seuraavalle yhdyskäytävälle. Reitityspäätös voi perustua tietoon internetyhteydellisestä yhdyskäytävästä, joka on verkon juurilaite HWMP-protokollan mukaisesti. Reitityspäätös voi myös perustua AODV-protokollan mukaisen reititystietoon yhdyskäytävästä, jolla on internetyhteys.

Ehdotusta on testattu laboratorio olosuhteissa kolmella reitittimellä. Testauksessa on käytetty laitteita, jotka tukevat Pycom LoPy- laiteohjelmistoa. PyCom on *LoRaWAN* yhteensopiva micropython alustasovellus. Testeissä ei ole huomioitu liikennöintiä kohti päätelaitetta tai tietoturvaan liittyviä kysymyksiä. Testatut yhdyskäytävät ovat toimineet vain yhden hajautuskertoimen eli yhden kanavan alaisuudessa.

3.2.3 Yhteenveto reitittävistä ratkaisuista

Reitittämisen edellytyksenä on jonkin ennalta määritellyn reititysprotokollan käyttäminen. Reititystä voidaan toteuttaa staattisilla tai dynaamisilla protokollilla. Staattiset reititysprotokollat vaativat manuaalista konfigurointia, kun taas dynaamiset reititysprotokollat käyttävät automaattisia algoritmeja, jotka määrittävät reitit datan välittämiseen verkossa. IoT-verkossa reitityspäätös voidaan myös muodostaa tarveperusteisesti, jolloin huomioidaan ajanhetkellä saatavilla olevat laitteet eli verkon topologia. Esimerkki tällaisesta protokollasta on AODV. Reititysprotokollan käyttäminen edellyttää ad hoc -verkoissa laitteiden jatkuvaa- tai ennalta määriteltyä saatavuutta eli päällä oloa. Vaatimus jatkuvasta saatavuudesta saattaa tehdä mahdottomaksi paristokäyttöisen päätelaitteiden hyödyntämisen reitittävänä laitteena virrankulutuksen takia. LoRa(WAN)-verkoissa voidaan käyttää useita eri hajautuskertoimia ja kanavataajuuksia yhtäaikaisesti. Mikäli reititysprotokolla käyttää useita yhtäaikaisia kanavataajuuksia tai hajautuskertoimia edellyttää tämä reititykseen osallistuvalta laitteelta monikanavaista radiomoduulia. Monikanavaisen radiomoduulin hinta on huomattavasti korkeampi verrattuna yksikanavaiseen radioon, joita pääsäätöisesti käytetään päätelaitteilla. Monikanavainen radio kuluttaa myös suhteessa enemmän virtaa sen ollessa käytössä.

Reitittäminen edellyttää verkolta sen topologian tuntemista, eli minkä laitteiden kautta saavutetaan haluttu lopputulos ja toisaalta reitittävien laitteiden saatavuutta reititystapahtuman ajanhetkellä. Kuten aiemmissa luvuissa on kuvattu, reitittämistä voidaan tehdä niin päätelaitteen kuin yhdyskäytävänkin avulla. Iot-mesh verkoissa käytettäviä reititysprotokollia ja näihin liittyviä teknisiä toteutuksia on useita ja näitä on esitelty lisää taulukon 3.2 julkaisuilla. Aiheen ympärillä tehdään jatkuvaa tutkimus- ja kehitystyötä parantaen olemassa olevia toteutuksia ja kehittäen uusia innovaatioita.

3.3 Yhteenveto mesh toteutuksista

Liikennöintimäärien ennalta-arvaamattomuus on tyypillistä mehs-toteutuksilla. Laitemäärät ja sovelluksen välitys- ja tai reititystapa osaltaan määräävät, kuinka usein yksittäisen laitteen on kyettävä käsittelemään viestiliikennettä, vaikka se ei olisi kohdennettu kyseiselle laitteelle. Samat viestit voivat päätyä eri reittejä pitkin samalle päätelaitteelle ja viestien käsittely edellyttää valveillaoloa. Paristokäyttöisillä

Taulukko 3.2: Reitittävät ratkaisut

Julkaisu	Tyyppi	Toteutustapa
Dias et al.ónio [7]	Reitittävä-päätelaite	LoRaWAN
Sartori et al.[35]	Reitittävä-päätelaite	LoRa
Haubro et al.[15]	Reitittävä-päätelaite	LoRa
Aslam et al. [2]	Reitittävä-yhdyskäytävä	LoRaWAN
Ma Dwijaksara et al. [10]	Reitittävä-yhdyskäytävä	LoRaWAN
Ebi et al. [11]	Reitittävä-päätelaite	LoRa
Lundell et al.[28]	Reitittävä-yhdyskäytävä	LoRa
Web-sivu [33]	Reitittävä-päätelaite Reitittävä-yhdyskäytävä	LoRa
Dias et al. [8]	Yhdyskäytävä Reitittävä-päätelaite	LoRaWAN

päätelaitteilla nämä voivat aiheuttaa ongelmia sovellussuunnittelun kannalta. Edellä esitellyt mesh-toteutukset soveltuvat huonosti käyttötapauksiin, missä päätelaitteen virrankulutuksella on merkitystä. Toinen hieman hankalammin todennettava huomio on verkon kapasiteetti laitemäärän kasvaessa. Julkaisuissa esitellyt käytännön toteutukset on tehty pienillä laitemäärillä ja näiden osalta jää epäselväksi toteutuksien skaalautuminen laitemäärien kasvaessa. Pro gradun käytännön sovellus osuudessa tarkasteltava Meshtastic sovelluskehikon maksimi päätelaitemäärät voivat olla 40 - 80 päätelaitteessa sovellusversiossa 1.3. Laitemäärään vaikuttaa käytetty hajautuskerroin ja tulvaperusteisen välitys-algoritmin hyppyjen maksimi määrä. Mitä suuremmat arvot ovat kyseessä, sitä pienemmäksi muodostuu päätelaitteiden määrä verkon kuuluvuusalueella.

Mikäli virrankulutukseen tai päätelaitemääriin liittyvät huomiot voidaan sivuttaa, tarjoaa mesh-ratkaisut kustannustehokkaan tavan laajentaa verkon peittoaluetta. Samalla se kasvattaa vikasietoisuutta tuottamalla dynaamisia ja vaihtoehtoisia tiedonvälityspolkuja saatavilla olevien laitteiden avulla.

3.3.1 LoRa

Lähtökohtaisesti *LoRa* on tarkoitettu osaksi *LoraWAN*:ia. *LoRa*:a varten on tarjolla valmiita sovelluskirjastoja, jotka mahdollistavat sovelluskehityksen näiden avulla.

Käyttötapauksissa missä *LoRa*:a hyödynnetään esimerkiksi reitityksen laajentamiseen [28] tarkoituksena on kuitenkin siirtää *LoraWAN* dataa. Pelkästään *LoRa*:n perustuvissa toteutuksissa tietoturvakysymykset jäävät heikommalle huomiolle. *LoRa* itsessään ei tarjoa mitään tietoturvaan liittyviä palveluita vaan tietoturvaominaisuudet on tuotettava kehitettävän sovelluksen toimesta.

3.3.2 LoRAWAN

LoRaWAN-spesifikaation määrittämät toimintamallit asettavat selkeitä reunaehtoja, joiden muokkaaminen johtaa väistämättä jonkin toisen määritellyn ominaisuuden heikentämiseen. Toimintamallit voidaan jakaa kolmeen luokkaan. Ensimmäinen on virrankulutus. Toinen on verkossa olevien laitteiden määrät ja tiedonvälityksen viiveet. Kolmas on tietoturva ja siirtokapasiteetti. Kuvassa 3.2 oleva oranssi ympyrä esittää pistettä minkä sijainti ei saisi muuttua kehitettäessä uusia standardin mukaisia *LoRaWAN*-ominaisuuksia.

Kuva 3.2: Toimintamallit

Kustannustehokkain tapa *LoRaWAN*-verkon kuuluvuuden eli peittoalueen kasvattamiselle on mahdollistaa datan välittäminen päätelaitteiden toimista. Tämä on kuitenkin ristiriidassa *LoRaWAN*:in spesifikaation ja tähtimäisen verkkotopologian kanssa. Ehdotetuissa toteutuksissa haasteeksi muodostuu virrankulutus ja laitemäärien kasvaessa viiveet ja siirtokapasiteetti.

4 Käytännön sovellus LoRa mesh-verkossa

Meshtastic on avoimen lähdekoodin projekti, joka hyödyntää *LoRa*-modulointia, luodakseen mesh-verkkoja pitkän kantaman viestintään. Meshtasticia käytetään pääsääntöisesti luomaan oma yksityinen tekstiviesti verkko, jossa laitteet voivat lähettää viestejä suoraan toisilleen.

Mestastics on suunnattu käyttötapauksiin, missä toimintaympäristö ei tarjoa tietoliikennepalveluita ja tarvitaan tapa ryhmäviestiä tekstiviestien avulla. Käyttötapaus esimerkkinä retkeily sijanneissa, minne ei oletusarvoisesti tarjota markkinaehtoisia tietoliikennepalveluita niiden vähäisen käytön takia. Sovellusta voidaan myös hyödyntää katastrofialueilla, missä tietoliikennepalveluita ei ole saatavilla esimerkiksi rikkoutuneen infrastruktuurin vuoksi. Ryhmäviestimen käyttöliittymänä toimii tablet tai puhelin ja tietoliikennepalvelut tuotetaan Meshtastic päätelaitteiden avulla. Näiden lisäksi päätelaitteet voivat tarjota omia sijanititietojaan edellyttäen, että päätelaite tarjoaa GPS-kyvykkyyden. Yhteys päätelaitteen ja käyttöliittymän välillä on WiFi- tai Bluetooth perusteinen ja Meshtastic-verkossa on vähintään kaksi päätelaitetta ja puhelin ja tai tablet käyttöliittymänä. Omien sovelluksien kehittäminen on mahdollista sovelluksen tarjoaminen kirjastojen ja API-palveluiden kautta.

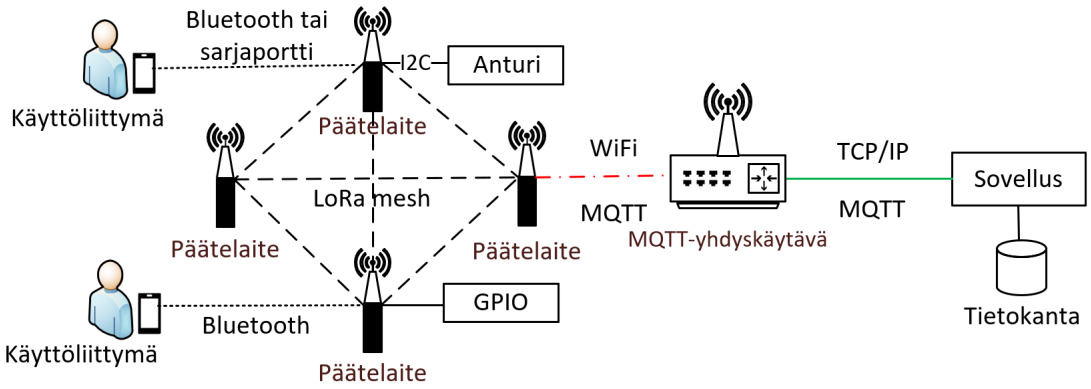
Koska kyseessä on vapaan lähdekoodin toteutus ja kehittäjäyhteisö dokumentoi tulosta joko Githubi:sa tai sovelluksen omilla web sivuilla, lähdeviittaamista ei ole tehty kuten aiemmin tässä dokumentissa. Sovellusta kehitetään jatkuvasti ja sen dokumentointi elää vastaavalla tavalla. Kaikki tässä luvussa ja sen aliluvuissa esitetty perustuu Mestastics dokumentaatioon sen eri muodoissa pro gradun kirjoitushetkellä [19].

4.1 Käytönnön toteutuksen esittely

Käytännön toteutuksessa tarkastellaan kuinka Mehstastic soveltuu yhtäaikaisesti tekstiviestipalvelu ja anturiverkko toimintaan, eli esimerkiksi lämpötilan tai ilmanpainetiedon mittaamiseen ja välittämiseen. Meshtastic esitellään tarkemmin alaluvusta 4.2 eteenpäin.

Käytännön toteutuksen toimintaperiaate esitellään kuvassa 4.1. Sovelluksen käyt-

töönottoon liittyy paljon konfigurointia ja tätä esteillään alaluvusta 4.6 eteenpäin. Sovellus toteuttaa tiedonsiirtoa *LoRa* mesh-verkossa ja luo *MQTT*-yhdyskäytävän internetin ja infrastruktuurittoman Meshtastic verkon välille. Toteutuksessa on myös mukana *GPIO*-ohjausta ja anturitiedon välittämistä. *MQTT*-viestiliikennettä voidaan tallentaa kehitetyn sovelluksen avulla tietokantaan myöhempää käyttöä varten. Luvussa 4.8 yhteenvedetään käytännön toteutuksen eri vaiheita.



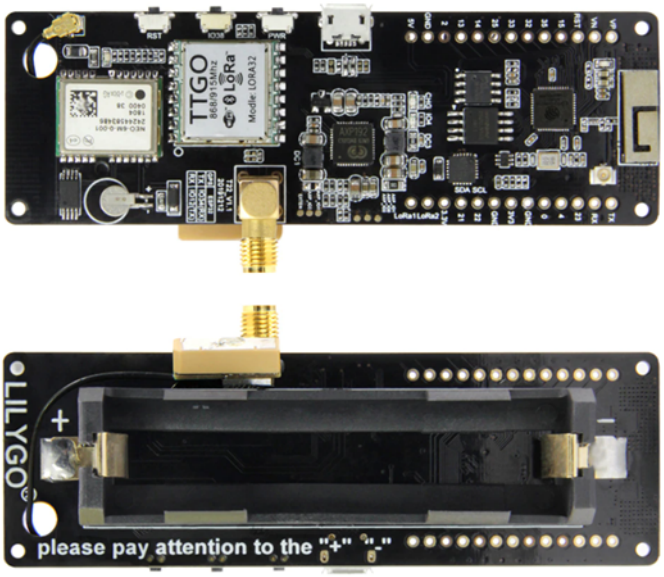
Kuva 4.1: Käytännön toteutus

4.2 Meshtastic päätelaite

Päätelaite muodostuu mikroprosessorista, joka on tyypiltään *ESP32* tai *nRF52* ja omaa *Bluetooth*- ja *WiFi*-kyvykkyyden. Näiden lisäksi päätelaitteella on *LoRa*-radio ja mahdollinen *GPS*-piiri. Päätelaitteen hinta on 25€:sta ylöspäin ja hintaan vaikuttaa toimituksen lisävarusteena olevat kotelointi, *GPS* ja tai *WiFi*-tuki, jotka eivät ole ydintoimintojen kannalta pakollisia. Toimituksen hintaan voi myös vaikuttaa laadukkaampi antenni ja ladattava akkuparisto. Mikä tahansa *ESP32* tai *RF52* ja *LoRa*-radio ei ole tuettuna. Tuetulle päätelaitetyypille on saatavilla valmiiksi käännetty laiteohjelmisto (engl.firmware). Kuvassa 4.2 esimerkki tuetusta päätelaitteesta.

4.3 Meshtastic verkon muodostuminen

Meshtastic tietoliikenneverkko muodostuu päätelaitteista, jotka yhdistetään keskenään IoS- tai Android sovelluksen avulla. Sovellus voidaan asentaa puhelimeen



Kuva 4.2: meshtastic päätelaite

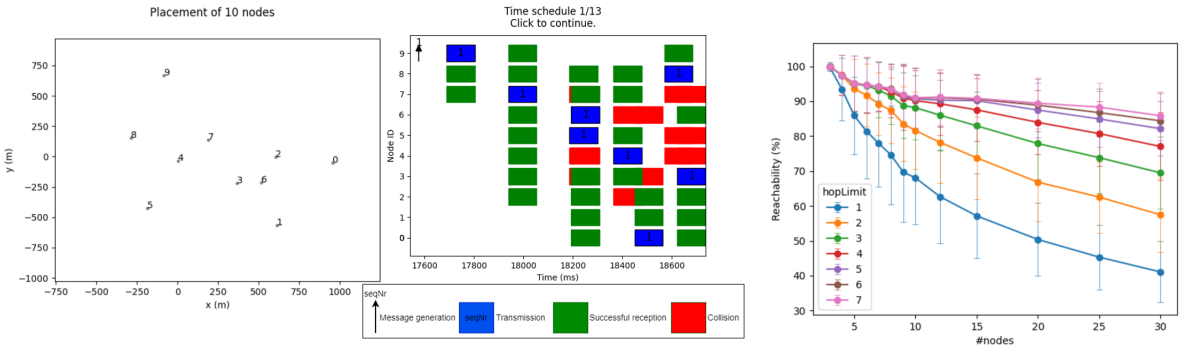
tai tabletille näiden käyttöjärjestelmärajoitusten puitteissa ja on saatavilla vastaavista sovelluskaupoista. Sovellus toimii verkon hallintapisteenä ja tuottaa samalla tekstiviesti- ja karttasovelluspalvelut verkkoon kytketyille käyttöliittymälaitteille. Jokainen Meshtastics-verkko on oma tietoturvallinen kokonaisuutensa. Kanavamuodostuksen yhteydessä päätelaitteella otetaan käyttöön yhteinen jaettu salaisuus, jonka avulla hyötykuorma salataan. Salauksessa käytetään AES-perusteista kryptografiaa. Salaamattoman verkon muodostaminen on myös tarvittaessa mahdollista. Tiedonsiirrossa noudatetaan *LoRa*-määrityksen mukaisia toimintatapoja symbolimuodostuksen, hajautuskertoimien, kanavataajuuksien ja FEC-toiminteiden osalta. Koska käytössä on *LoRa* ja sen käyttämät kanavataajuudet, ovat maakohtaiset ISM-määritykset voimassa rajoituksineen.

4.3.1 Verkon toimintaperiaate

Meshtastic-verkossa päätelaitteet välittävät saamansa viestin uudelleen toisille päätelaitteille. Oletuksena viesti välitetään tulvimisperiaatteella kolme kertaa eri aikaikkunoissa tiedonsiirron varmistamiseksi ja jos vastaanottava päätelaite on jo kuullut viestin, hylkää se sen automaattisesti. Tulvimista kontrolloidaan kehyksessä olevalla maksimivälitysmäärä tiedolla. Jokainen välitys vähentää kehyksen otsikossa olevaa välitysmääräarvoa yhdellä, kunnes se saavuttaa arvon nolla jolloin viestiä ei

enää välitetä.

Verkon toimintaa on mahdollista simuloida tarkoitusta varten luodun sovelluksen avulla. (kuva 4.3). Simuloinnin avulla voidaan tarkastella päätelaite- ja hyppymäärien kuten myös etäisyyksien vaikutusta verkon toimintaan. Simulointi kykenee myös tuottamaan graafeja testiajo perusteisti liikennöinnin todennäköisyyden toteutumisesta määritellyillä päätelaite- ja välitysmäärillä.



Kuva 4.3: meshtastic simulaattori

Meshtastic-verkon algoritmi on saanut vaikutteita Radiohead [29] sovelluskehityskehikosta ja siellä olevasta *LoRa*-mesh kirjastosta. Versiossa 1.3 Meshtastic ei kontolloi välityspäätöksiä, verkossa olevien laitteiden tilaa tai niiden saatavuutta muuta kuin datagrammissa olevan pakettilaskurin avulla.

4.3.2 Meshtastic tiedonsiirto

Meshtasticin tiedonsiirron toiminnan voi jakaa kahteen osaan. Perustana on *LoRa*-radiokerros, joka vastaa symbolimuodostuksesta ja radiokontrollista. Jokaiselle tuetulle *LoRa*-radiotyypille on oma sovelluskirjastonsa, jonka avulla symbolikäsittelyä hallinnoidaan. Kehysmuodostuksessa kättely on 32,25 viserryksen mittainen, josta 28 on nousevaa ja 4,25 laskevaa. Kättely vastaa samaa tarkoitusta, mitä *LoRaWAN*-verkoissa on aiemmin esitelty, mutta on viserryksien kannalta huomattavasti pidempi.

Pakettikerros muodostaa dataa radiokerrokselle aliluvussa 4.5.1 esitellyn protokollapuskureiden avulla ja pakettimuodostuksen voi jakaa kolmeen vaihtoehtoon.

- Varmentamaton tiedonsiirto, ei monihyppyä. Tässä mallissa dataa välitetään

tulvintaperiaatteella, välittämättä onko jokin päätelaite vastaanottanut sitä vaiko ei. Vastaanottanut päätelaite ei välitä saamaansa pakettia uudelleen.

- Varmennettu tiedonsiirto, ei monihyppyä. Malli vastaa edellä kuvattua, sillä erotuksella, että vastaanottajan tule kuitata paketti saapuneeksi ACK-viestillä.
- Monihyppy tulvinta. Kehyksessä ilmoitetaan välityskertojen maksimimäärä ja käytössä on myös ACK-kuittaus, jonka avulla tulvimista voidaan kontrolloida uudelleenlähetyksien osalta.

Edellä kuvatuille malleille on yhteistä *CSMA/CA* (engl. Carrier-Sense Multiple Access with Collision Avoidance) kanavavarauksen käyttäminen. Lähettäjän on tehtävä kanavan aktiivisuuden tarkistus ennen lähettämistä. Mikäli kanava on varattuna, odotetaan sen vapautumista. Kanavan vapauduttua lähettäjä odottaa satunnaisen ajan ennen lähetyksen aloittamista, jonka jälkeen tarkistetaan uudelleen kanavan vapaana olo. Jos kanava on edelleen vapaana, suoritetaan lähetys.

4.3.3 Kehys

Kehys rakentuu taulukossa 4.1 esitetyn mallin mukaisesti. Taulukosta on havaittavissa, että hyötykuorman maksimi koko on 237 tavua. Kehyksen otsikko-osat ovat aina salaamattomia, kun taas hyötykuorma on oletusarvoisesti salattua. Ominaisuus mahdollistaa tiedon välittämisen ilman datan purkua pelkän *LoRa*-radion avulla. Noodi-id muodostetaan päätelaitteen Bluetooth mac-osoitteen viimeisestä 4 tavusta ja käytössä olevan *LoRa*-radio valmistajan IEEE:n määrittämästä OUI-arvon alimmasta 3 tavusta. Edellä kuvatun perusteella voidaan päätellä, että jokainen Meshtastic päätelaite voi välittää dataa, vaikka ei olisi paritettuna Meshtastic verkossa. Päätelmän perusteena on samojen kanavataajuuksien, hajautuskertoimien ja virheenkorjaus arvojen käyttäminen.

4.4 Meshtastic verkonhallinta

Käyttöliittymäsovelluksen avulla voidaan hallita verkkoon liittyviä asetuksia kuten hajautuskerrointa ja ennakoivan FEC-virheenkorjauksen pariteettibittien määriä. Arvoja voidaan muuttaa ennalta määriteltyjen sovellusnimikkeiden avulla, joita kuvataan taulukossa 4.2. Omien hajautuskerroin, FEC-, salaus-, kanavataajuusarvo-

Taulukko 4.1: Datagrammi [19]

Pituus	Tyyppi	Käyttötarkoitus
1 tavu	Kokonaisluku	Synkronointi. Aina 0x28
4 tavua	Kokonaisluku	Paketin otsikko. Vastaanottajan node-id 0xFFFFFFFF broadcast osoite
4 tavua	Kokonaisluku	Paketin otsikko. Lähettäjän node-id
4 tavua	Kokonaisluku	Paketin otsikko. Lähettäjän yksiköllinen paketti-id
32 bittiä	Bitti	Paketin otsikko. Välityskertojen määrä. Pakettikuittaus (ACK). Tässä om vapaana bittejä muuhunkin käyttöön
1 - 237 tavua	Tavu	Hyötykuorma. Käyttämättömiä tavuja ei lähetetä

jen määrittäminen on myös mahdollista komentokehoitteen avulla. Aihetta käsitellään tarkemmin alaluvussa 4.4.1

4.4.1 Päätelaitehallinta

Päätelaitteilla on monia parametrejä, joita pystyy hallitsemaan joko erikseen asennettavalla komentokehotteella tai tarkoitusta varten luodulla web-sovelluksella. Python perusteinen komentokehote kommunikoi päätelaitteen kanssa sarjamuotoisen liikennöinnin avulla. Hallintaa voidaan myös tehdä TCP/IP:n avulla ja tässä tapauksessa päätelaitteessa on oltava *WiFi*-ominaisuus aktiivisena. Sarjamuotoinen yhteys voidaan muodostaa USB-kaapelin avulla, joka samalla toimii myös päätelaitteen virtalähteenä ja/tai lataa siellä olevaa akkuparistoa. Komentokehote asennetaan Python PIP-pakettihallinnan avulla. Käytettäessä PIP-pakettihallintaa koneella on oltava asennettuna Python versio 3.4 tai uudempi. Asentaminen suoritetaan komennolla.

```
pip install --upgrade meshtastic
```

Osoitteesta `https://client.meshtastic.org/` on saatavilla web-sovellus, joka mahdollistaa päätelaitehallinnan graafisesti. Web-sovellus tukee *Bluetooth*, *HTTP*- ja sarjamuotoista liikennöintiä ja vastaa ominaisuuksiltaan edellämainittua PIP-paketista asennettavaa päätelaitehallintasovellusta.

Päätelaitteen asetuksien muuttaminen tapahtuu käyttämällä asennettua komentokehotetta tai web-sovellusta. Seuraava komento muuntaa päätelaitteen välittävä-

Taulukko 4.2: Käyttöliittymä sovelluksen oletus kanavat [19]

Nimi	Nopeus kbps	Hajautuskerroin	FEC CR	Kanavataajuus kHz
Short Fast	18,89	7	4/5	500
Short slow	4,69	7	4/5	125
Medium fast	1,2	10	4/6	250
Medium slow	0,75	11	4/7	250
Long fast	0,19 (oletus)	9	4/8	31
Long slow	0,13	12	4/8	125
Very long slow	0,04	12	4/8	31

päätelaite muotoon ja sammuttaa päätelaitteella mahdollisesti olevan näytön, *WiFi*- ja *BLE*-radion.

```
meshtastic --set device.role ROUTER
```

Päätelaitehallinta jakaantuu useisiin eri hallintalohkoihin, josta verkkoarkkitehtuurin kannalta mielenkiitoisimmat löytyvät *Device*-, *LoRa*- ja *Power*-lohkojen alaisuudesta. Meshtastic komennon formaatista voi huomata **–set** operaattorin, jonka avulla määritellään, että kyseessä on muutos. Vastaavasti operaattorin **–get** avulla voidaan pyytää tietoa. Operaattoria seuraa lohko ja piste notaatiolla lohkossa oleva funktio. Viimeisenä on parametri, joka halutaan tarkastella tai ottaa käyttöön.

Device-konfiguraatiossa voidaan ottaa kantaa, miten päätelaite käyttäytyy meshtilanteissa. Alaluvussa 4.3.2 kuvattiin kolme eri välitystapaa ja täällä tehtävän konfiguraation avulla voidaan päätelaite määrittää käyttämään yhtä näistä.

LoRa-konfiguraatiossa voidaan vapaasti ottaa kantaa siihen minkälainen hajautuskerroin, virheenkorjaus ja kanavataajuus on käytössä päätelaitteen oletuskanavalla. Päätelaite voi kuulua yhtä aikaa useammalle kanavalle, mutta näiden *LoRa*-määritykset vastaavat oletuskanavan arvoja.

Power-konfiguraatiossa voidaan määrittää kuinka nopeasti *CPU*- tai *LoRa*-radio kytketään pois päältä. Täällä voidaan myös ottaa kantaa siihen, kuinka nopeasti siirrytään eri unitiloihin virransäästämiseksi.

4.5 Sovelluskehitys meshtastic alustalla

Avoin lähdekoodi mahdollistaa kaikkien sovellusosien kehittämisen. Alaluvuissa ei käsitellä laiteohjelmisto (engl. firmware), protokollapuskureiden tai käyttöliittymän sovelluskehitystä lainkaan. Näidenkin kehittäminen on mahdollista omien tarpeiden mukaisesti. Muuten sovelluskehityksen voi jakaa kahteen osaan.

1. Kehitettyä sovellusta suoritetaan suoraan päätelaitteella ja puhutaan ns. moduli perusteisesta kehittämisestä. Meshtastic-päätelaitteilla on valmiina moduleita, esimerkiksi *MQTT*- tai *GPIO*-ominaisuuksien hyödyntämistä varten. Modulit toimivat siltana päätelaitteen verkkopalveluiden ja ulkoisten sovellusten välillä.

2. Meshtastic-verkkoa ja sen päätelaiteita käytetään joko *HTTP*, *Bluetooth* tai sarjaliikenne perusteisesti. Tässä mallissa sovellus toimii päätelaitteen ulkopuolisena ja päätelaitetta käytetään välittämään sovelluksen datavirtaa verkon ja sovelluksen välillä.

Edellä kuvatut mallit eivät ole toisiaan poissulkevia. Esimerkiksi päätelaiteeen *GPIO*-palvelun käyttäminen edellyttää modulin aktivointia. Tiedonsiirto modulin ja ulkoisen sovelluksen välillä voi olla *HTTP*, *Bluetooth* tai sarjaportti perusteista. *WiFi*- ja *Bluetooth*-yhteyden yhtäaikainen käyttäminen ei ole mahdollista. Vastaavasti *WiFi* ja sarjaliikenteen käyttäminen on.

4.5.1 Protokollapuskurit

Meshtastic-sovelluksen sisällä tietoa käsitellään ja välitetään protokollapuskureiden (engl. Protocol buffers) avulla [14]. Protokollapuskureiden toimintalogiikka perustuu tietomallin muodostamiseen. Tietomalli käännetään valitulla ohjelmointikielellä sovelluksessa käytettäväksi protokollapuskuriksi kirjoitus- ja lukuoperaatioita varten. Kyseessä on laitteisto ja ohjelmointikieli riippumaton ohjelmointikehikko, joka on vapaasti saatavilla Googlen tuottamana. Meshtastic-sovelluksessa edellä kuvattu protokollapuskuri nimetään protobuff nimellä ja jokaisella sovelluksen osalla on oma tietomallimäärityksensä tiedonvälitystä varten.

4.5.2 Moduulikehitys

Kehiteyn moduulin tulee vähintään olla Meshmodule luokan instanssi, jolloin se perii tiedonkäsittelyyn tarvittavat ominaisuudet ja se rekisteröityy automaattisesti käsittelemään verkossa liikkuvia datapaketteja. Moduuli-hierarkiaa voidaan kuvata seuraavasti.

MeshModule pääluokka. MeshModule (/src/mesh/MeshModule.h) sisältää tiedon lähettämiseen ja vastaanottamiseen liittyvät funktiot. Tämä luokka välittää datan radiomoduulille.

SingePortModule perii MeshModulen ominaisuudet. Moduuli (src/mesh/-SingePortModule.h), jonka avulla voidaan lähettää ja vastaanottaa porttinumero perusteisesti dataa. Meshtastic-verkossa välitetyissä viesteissä on aina porttinumerotieto, jolloin datan käsittely voidaan ohjata tietylle moduulille.

ProtobufModule perii SingePortModule ominaisuudet. Moduuli (srv/mesh/-ProtobufModule.h), missä dataa voidaan käsitellä protobuff eli protokollapuskureiden avulla.

Kehittäminen tapahtuu pääsääntöisesti luomalla uusi instanssi luokasta SinglePort-Module mikäli dataa halutaan käsitellä bittitasolla tai ProtobufModule jos dataa halutaan käsitellä protokollapuskureiden avulla.

4.5.3 Ulkopuolinen sovelluskehitys

Sovelluskehitystä voidaan tehdä esimerkiksi Python SDK:n avulla. Datan lähettäminen Meshtastic-verkkoon voidaan toteuttaa yksinkertaisimmillaan ohjelmakoodi 4.1 avulla, joka on suora lainaus kehittäjäohjeista [19]. Esimerkkiohjelmassa ladataan riveillä 1-6 tarvittavat kirjastot ja muodostetaan sarjaporttiyhteys päätelaitteeseen. Rivillä 9 lähetetään 'Hello Mesh' viesti oletuskanavalle. Riveillä 11-16 pyydetään kytketyltä päätelaitteelta käytössä olevat parametriarvot ja muutetaan niitä GPS-päivitystaajuuden osalta. Rivillä 17 tulostetaan voimassa olevat arvot. Rivillä 19 tallennetaa konfiguraation päätelaitteeelle ja rivillä 21 suljetaan sarjaliikenneyhteys.

Ohjelmakoodi 4.1: Esimerkki Python sovellus

```
1 import meshtastic
```

```
2 import meshtastic.serial_interface
3
4 # By default will try to find a meshtastic device,
5 # otherwise provide a device path like /dev/ttyUSB0
6 interface = meshtastic.serial_interface.SerialInterface()
7
8 #SendData to send binary data, see documentations for other options.
9 interface.sendText("hello mesh")
10
11 ourNode = interface.getNode('^local')
12 print(f'Our node preferences:{ourNode.radioConfig.preferences}')
13
14 # update a value
15 print('Changing a preference...')
16 ourNode.radioConfig.preferences.gps_update_interval = 60
17 print(f'Our node preferences now:{ourNode.radioConfig.preferences}')
18
19 ourNode.writeConfig()
20
21 interface.close()
```

4.6 Sovelluksen konfiguraatio

Seuraaviin osiin tulee kiinnittää huomiota Meshtastic käyttöönoton ja käytön aikana. Päätelaitetta ei saa käynnistää ilman antennia, koska se saattaa vaurioittaa LoRa-radion pysyvästi. Päätelaitteelle konfiguroidut arvot ovat selkokielisinä esimerkiksi WiFi- tai MQTT-salasanan osalta. Kanavilla käytetty salausavaimet ovat saatavilla suoraan päätelaitteelta selkokielisinä. Edellä mainittujen tietojen lukemiseen riittää, että päätelaitteen saa kytkettyä sarjakaapelilla koneelle, joka kykenee suorittamaan Python komentokehotetta.

Käyttöliittymäsovelluksien ja päätelaitteiden käyttöönoton yhteydessä tulee kaikkiin käytettäviin laitteisiin asentaa sama ohjelmistoversio. Päätelaitteet toimitetaan valmiiksi asennetulla laiteohjelmistolla (engl. fimware) ja korkealla todennäköisyydellä toimitettu versio ei ole yhteensopiva muiden ohjelmistojen kanssa. Ohjelmistoversioiden poikkeavuudet päätelaitteiden ja käyttöliittymäohjelmistojen välillä aiheuttavat virheellistä toimintaa sovelluksella.

Komentokehote asentuu viimeisimpään saatavilla olevaan versioon käytettäessä Pythonin PIP-pakettihallintaa. Asennus on kuvattu alaluvussa 4.4.1.

Dokumentin kirjoitushetkellä käyttöliittymäsovellus asentuu versioon 1.2 ja kun taas päätelaitteiden ohjelmisto asentuu versioon 1.3. Käyttöliittymäsovelluksen versio 1.3 on mahdollista asentaa sovelluskaupasta liittymällä Meshtastic testiohjelmaan osoitteesta.

`https://play.google.com/apps/testing/com.geeksville.mesh/join?hl=en-US`

Päätelaitteiden laiteohjelmiston (engl. firmware) päivittäminen suoritetaan meshtastic-flasher ohjelmistolla, joka tarjoaa graafisen käyttöliittymän toimenpidettä varten. Sovellus asennettaan PIP-paketista kuten aiemmassa alaluvussa 4.4.1 kuvattu komentokehotekin.

```
pip install meshtastic-flasher
```

Flasher sovellus noutaa viimeisimmät saatavilla olevat laiteohjelmistotiedostot, tunnistaa päivitettävän päätelaitetyypin ja päivittää siihen noudetun laiteohjelmiston. Päätelaite kytketään sarjaporttiyhteydellä siihen tietokoneeseen, missä meshtastic-flasher ohjelmistoa suoritetaan.

4.6.1 Meshtastic kanavan käyttöönotto

Ohjelmistoasennuksien jälkeen voidaan muodostaa ensimmäinen tietoliikennekanava. Yksinkertaisimmillaan kanava muodostetaan parittamalla käyttöliittymä- ja päätelaite keskenään noudattamalla Bluetoothista tuttua numerosarjaan perustuvaa paritus tapaa. Parittamisen voi suorittaa käyttöliittymäsovelluksesta etsimällä Bluetooth laitetta ja valitsemalla päätelaitteen-id:n, joka esiintyy päätelaitteen näyttöruudulla. Paritus vahvistetaan syöttämällä käyttöliittymään päätelaitteella esitettävä numerosarja. Ohjelmistoversiossa 1.3 ensimmäinen tietoliikennekanava muodostuu automaattisesti ja on tyypiltään taulukon 4.2 mukaisesti *long fast*. Mikäli Meshtastic verkossa on useampi pääte- ja käyttöliittymälaite, voidaan tekstiviestien välittäminen alkaa heti tällä oletuskanavalla. Oletuskanava jakaa oletus salaisuuden tarkoittaen, että kaikki Meshtastic verkon peittoalueella olevat päätelaitteet pystyvät kuulemaan ja purkamaan kanavan viestinnän.

Kanavahallinnassa on huomioitava, että kanavan *LoRa*-arvot ja jaettu salaisuus vastaavat toisiaan kaikilla päätelaitteella. Kun hallintaa tehdään käyttöliittymäsovellukselta jaettu salaisuus muodostuu automaattisesti. Muissa tapauksissa jaettu salaisuus voidaan tarvittaessa antaa omana arvonaan. Käyttöliittymälaite kykenee hallitsemaan yhtä päätelaitetta kerrallaan Bluetooth yhteyden avulla. Tietokoneen

komentokehotteen kautta voidaan hallita yhtäaikaisesti useampia päätelaitteita ja määrittävänä tekijänä on USB-porttien määrä.

4.6.2 Oletuskanavan muuttaminen

Oma kanava voidaan määrittää valitsemalla käyttöliittymäsovelluksen kuvassa 4.4 esittämä lukon kuva. Lukkovalinnan jälkeen kanava voidaan nimetä uudelleen ja sille voidaan asettaa taulukon 4.2 mukaisesti *LoRa*-arvot alasvetovalikosta. Muut käyttöliittymälaitteet voivat liittyä kanavalle lukemalla kuvassa näkyvän qr-koodin. Koodi voidaan myös lähettää esimerkiksi sähköpostiviestinä toiselle käyttöliittymälaitteelle, joka on yhteydessä päätelaitteeseen.

Kuva 4.4: Kanavahallinta

Kanavahallintaa voidaan myös tehdä aiemmin alaluvussa 4.4.1 mukaisesti päätelaitteen komentokehotteen avulla tai web-sovelluksella, jolloin hallinta voi olla yksityiskohtaisempaa. Käyttöliittymäsovellus on suunnattu loppukäyttäjälle, jolla ei ole tietoa tai kiinnostusta siihen miten eri konfigurointi arvot vaikuttava verkon toimintaan. Tyhjän päätelaitteen pystyy kytkemään kanavalle komentokehotteella komennolla:

```
meshtastic --seturl http://xxx/?xxx
```

Edellä kuvattu http://xxx/?xxx edustaa kuvassa 4.4 olevaa QR-koodin tietosisältöä. URL-tieto voidaan jakaa kuvassa näkyvän jako-ikonin avulla esimerkiksi sähköpostiviestinä.

Meshtastic kanavat jakavat *LoRa*-tiedonsiirron arvot. Mikäli oletuskanavalla on otettu käyttöön esimerkiksi taulukon 4.2 mukaisesti *medium slow* kanavamääritys kaikki muutkin kanavat käyttävät oletuskanavan arvoja. Kanava numero 1 on pääkanava ja samalla myös oletuskanava ja tätä ei voi muuttaa. Kanava voi olla aktiivinen tai se voi olla poistettu käytöstä, mutta kaikkien käytettävien kanavien tulee olla jatkuvassa numeerisessa järjestyksessä. Esimerkiksi kanava 4 ei voi olla aktiivinen ilman, että kanava 3 ei myös olisi aktiivinen. Jokaisella kanavalla voi olla oma AES-salaisuutensa ja omien avaimien käyttäminen on myös mahdollista. Salauksen käyttö on kanavakohtainen asetus ja salaukseen käyttäminen ei ole pakollista. Kanavakohtaisia asetuksia ovat myös esimerkiksi määritys välitetäänkö kanavan viestiliikenne *MQTT*-yhdyskäytävälle ja koskeeko välittäminen sisään vai ulos lähtevää dataliikennettä. *GPIO*-ominaisuuden käyttäminen on myös kanavakohtainen asetus. Kanava nimissä on muutamia varauksia kuten *gpio* ja *admin*. *Admin*-kanava on tarkoitettu päätelaitteiden etäältä tapahtuvaa hallintaa varten ja *gpio* on tarkoitettu päätelaitteella olevien digitaaliliityntöjen kontrollointia varten.

Komentokehotteella kanavia käsitellään perustuen indexinumeron ja numerointi lähtee liikkeelle ohjelmoinnista tutulla arvolla 0, joka vastaa kanavanumeroa 1. Muodostettaessa kanavaa tarvitaan globaali uniikki-id, kanavan nimi ja kanavaan liittyvä salausmääritys. Seuraavat komennot muodostavat **My Channel** nimisen kanavan, jonka globaali-id on 1234 ja se on salattu järjestelmän määrittämällä 256-bittisellä salausavaimella.

```
meshtastic --ch-set id 1234 --ch-index 0
meshtastic --ch-set name "My Channel" --ch-index 0
meshtastic --ch-set psk random --ch-index 0
```

Salauksen voi ottaa pois päältä komennolla.

```
meshtastic --ch-set psk none --ch-index 0
```

Oman salausavaimen voi määrittää alla olevalla komennolla ja avain pitää olla joko 128 tai 256 bittiä pitkä.

```
meshtastic --ch-set psk 0x1a1a1a1a2b2b2b2b1a1a1a1a2b2b2b2b
1a1a1a1a2b2b2b2b1a1a1a1a2b2b2b2b --ch-index 0
```

4.6.3 Meshtastic WiFi

Päätelaitteen WiFi-parametrien muuttaminen voidaan tehdä komentokehotteelta tai web-sovelluksen avulla. Päätelaite voi toimia itsenäisenä tukiasemana tai se voidaan liittää osaksi olemassa olevaa WiFi-verkkoa. Itsenäisen tukiasemamoodin saa päälle komentokehotteesta komennolla.

```
meshtastic --set wifi_ap_mode true
```

Päätelaite liitetään osaksi olemassa olevaa WiFi-infraa määrittämällä verkon SSID eli nimi ja kytkeytymiseen liittyvä salasana. Komentokehotteella arvot annetaan seuraavilla komennoilla.

```
meshtastic --set "wifi_ssid" XXXX
meshtastic --set "wifi_password" XXXX
```

Mikäli salasanoissa tai verkon nimi tiedoissa on välilyöntejä, ilmoitetaan tiedot lainausmerkkien sisällä. Päätelaitteen ip-numeron saa selville päätelaitteen näyttöruudulta onnistuneen verkkoon liittymisen jälkeen.

WiFi-aktivointi mahdollistaa käyttöliittymäsovelluksen kytkeytymisen päätelaitteelle tcp-ip verkon kautta. Käyttöliittymän ja päätelaitteen pitää olla samassa ip-avarauudessa.

4.6.4 MQTT

MQTT on tiedovälitysprotokolla, joka tuottaa kevyen, yksinkertaisen ja avoimen kommunikointitavan sovellusten välille. *MQTT* hyödyntää tiedonvälityksessä pääsääntöisesti TCP/IP protokollaa, joka tarjoa häviöttömän, kaksisuuntaisen tavan välittää tietoa. *MQTT*-toimintatapa perustuu julkaisija-tilaaja malliin, missä jokin laite ja siellä oleva ohjelmisto julkaisee aiheen (engl. publish) ja toiset laitteet ja siellä olevat ohjelmistot voivat tilata (engl. subsrcibe) julkaistun datan. Tilauksien ja julkaisuiden kesipisteenä on *MQTT*-yhdyskäytävä, jota kutsutaan myös välittäjäksi (engl. broker). Julkaisija julkaisee aiheen (engl. topic) välittäjälle ja kuluttaja tilaa aiheen välittäjältä. Toimintatapa mahdollistaa kommunikointimallin, missä yksittäinen viesti voidaan lähettää kaikille aiheen tilanneille sovelluksille yhtäaikaisesti. Viestin välittäminen ilman välittäjän tuottamia palveluita ei ole mahdollista ja välittäjä huolehtii myös tietoturvaan liittyvistä kysymyksistä käyttäjätunnistamisen ja sertifikaattien avulla [16].

Julkaistava aihe voi olla yksitasoinen tai se voi muodostaa hierarkian. Esimerkkinä aihe, joka sisältää useiden päätelaitteiden tuottamaa mittaustietoa. Jokainen

päätelaite voi esiintyä aiheen sisällä itsenäisenä objektina ja kaikkien päätelaitteiden mittaustulokset saa tilaamalla em. aiheen. Vastaavasti on mahdollista tilata aiheesta vain yksittäisen päätelaitteen data [16].

MQTT-hyötykuorman koko voi olla jotain 2 bitin ja 256 MB välillä. Hyötykuorman formaattia ei ole määritelty ja binääridatan lähettämien on myös mahdollista. Hyötykuorman sisällöstä ja formaatista vastaa viimekädessä sovelluksen kehittäjä [18]. Useasti *MQTT*-hyötykuorman sisältö on formaatiltaan *JSON*-muotoista.

JSON [12] (engl. JavaScript Object Notation) on avoin, kevyt, ohjelmointikieli riippumaton, tekstiperusteinen tiedon esitys formaatti. Formaatin avulla sovellukset pystyvät yksinkertaisesti muodostamaan tai tulkitsemaan välitettyä dataa. Ihmiselle formaatin tulkitseminen on myös verrattain helppoa sen esitystavan johdosta. JSON määrittää datan esitystavan yksinkertaisella objekteihin ja elementteihin perustuvalla tavalla. *JSON*-objekti muodostuu kaarisulkeista ja sen sisällä esitettävissä elementeistä. Elementit ovat avain- ja arvo-pareja, jotka esitetään lainausmerkeissä kaksoispisteellä erottaen.

```
{"avain": "arvo", "toinen_avain": "toinen_arvo"}
```

JSON-objektin sisällä voi olla toinen *JSON*-objekti tai objekti voi muodostua listasta objekteja

```
{"avain": "arvo",
osoite : [{"postinumero": 1234},
          {"postinumero": 25677}]}
```

Kun sovellukselle toimitetaan dataa *JSON*-formaatissa mahdollistetaan datan käsittely objektissa olevien avaintietojen perusteella.

MQTT-yhdyskäytävä ei varastoi viestiliikennettä vaan olettaa, että tilaajat kykenevät vastaanottamaan viestin. Malli soveltuu huonosti käyttötapauksiin, missä vastaanottava laite voi mennä virransäästötilaan. *MQTT*-protokollasta on julkaistu *MQTT-SN*-laajenne, missä yhdyskäytävä varastoi välitettäviä viestejä. Tämän avulla viestejä voidaan välittää päätelaitteen ollessa valveilla.

4.6.5 MQTT-yhdyskäytävän konfiguraatio

Mosquitto on avoimenlähdekoodin *MQTT*-yhdyskäytävä eli välittäjä ja on saatavilla kaikille yleisesti käytössä oleville käyttöjärjestelmille. Asentaminen on suoraviivainen ja välitäjää voidaan suorittaa hyvinkin yksinkertaisella konfiguraatiolla. Oletuksena Mosquitto ei välitä liikennettä kuin osoitteessa 127.0.0.1. Koneen ulkopuoli-

nen liikenne ei siis ole sallittua ja käyttäjätunnuksia ei myöskään ole määritelty. Seuraavassa määritellään minimi konfiguraatio, joka avaa *MQTT*-oletusportin ja määrittää tunnuksettoman käyttöoikeuden kaikille julkaisuille ja tilauksille. Mosquitto hakemistosta löytyy tiedosto **mosquitto.conf** ja tiedostosta pitää editoida seuraavia kohtia. Sallitaan ja määritellään parametrit.

```
allow_anonymous true
listener 1883 x.x.x.x
```

x.x.x.x kuvaa suorittavan koneen ip-numeroa. Edellä kuvattua konfiguraatiota ei pidä missään tapauksessa käyttää sellaisenaan julkisena palveluna. Jokaiselle tilaukselle ja julkaisulle tulee määrittää vähintään käyttäjätunnus salasana pari, jonka avulla sitä voi kutsua. *MQTT*-yhdyskäytävän voi käynnistää alla olevalla komennolla.

```
mosquitto.exe -c mosquitto.conf -v
```

Parametri -c määrittää palvelimen käyttämään aiemmin kuvattua konfiguraatio tiedostoa ja -v määrittää suorituksen logidatan tulostuksen konsolille.

Yhdyskäytävän julkaisuja voi yksinkertaisimmillaan seurata Mosquitton mukana toimitettavalla asiakasohjelmalla. Julkaisun voi tilata komennolla.

```
mosquitto_sub.exe -h x.x.x.x -t "#" -v
```

Komennon parametri -h x.x.x.x määrittää *MQTT*-yhdyskäytävän ip-numeron. Parametri -t määrittää tilattavan julkaisun hierarkiarakenteen. # merkin avulla voidaan tilata kaikki julkaisut yhdyskäytävältä. Vastaavasti julkaisutilauksella /mesh/?/json/ tarkasteltaisiin mitä tahansa hierarkiaa, joka alkaa merkistöllä /mesh/ ja voisi sisältää jonkin merkin tai numeron a-z tai 0-9 kysymysmerkin tilalla. Tilauksen hierarkia päättyisi /json/ nimeillä. Parametri -v tulostaa kaiken logidatan konsolille. Asiakasohjelmiston avulla on yksinkertaista seurata viestiliikennettä kehitysvaiheessa.

4.6.6 Meshtastic päätelaitteen MQTT-konfiguraatio

Meshtastic päätelaite voi toimia kaksisuuntaisena yhdyskäytävänä *LoRa*-verkon ja *MQTT*-välittäjän välillä. Ominaisuus mahdollistaa kanavaliikenteen julkaisemisen tai vastaavasti kanavalle voidaan tilata eli välittää dataa *MQTT*:n avulla. Jokainen Meshtastic kanava toimii itsenäisenä, eli kanavalle voidaan määrittää lähetys ja tai vastaanotto *MQTT* avulla. Käyttöönotossa päätelaitteella aktivoidaan *MQTT*-moduuli ja konfiguroidaan tarvittavat asetukset. Toiminnallisuuden käyttäminen edellyttää

päätelaitteen kytkemistä *WiFi*-verkkoon ja *MQTT*-välityspalvelimen tietojen konfigurointia. Konfiguroinnin jälkeen kaikki Meshtastic kanavalla oleva liikenne välitetään tai vastaanotetaan kanavalla. Mikäli *MQTT*-yhdyskäytävä on otettu käyttöön alaluvun 4.6.5 mukaisesti, voi päätelaitteen *MQTT*-tuen ottaa käyttöön konsolilla seuraavilla komennoilla.

```
meshtastic --set mqtt.enabled true
meshtastic --set mqtt.address x.x.x.x
meshtastic --set mqtt.encryption_enabled false
meshtastic --set mqtt.json_enabled true
meshtastic --ch-index 0 --ch-set uplink_enabled true
```

Konfiguraatiossa käyttöönotetaan *MQTT*-tuki ja määritellään tähän liittyvän yhdyskäytävän ip-numero. Konfiguraatio määrittää myös, että dataa välitetään ulospäin. Mikäli dataa halutaan välittää Meshtastic verkkoon, tehdään se komennolla.

```
meshtastic --ch-set downlink_enabled true --ch-index 0
```

Edellä olevasta konfiguraatiosta on hyvä mainita *JSON*-tuen käyttöönotto, joka helpottaa myöhemmässä vaiheessa tehtävää datan käsittelyä.

Seuraavilla komennoilla voidaan ottaa kantaa *MQTT*-yhdyskäytävän tunnistautumiseen

```
meshtastic --set mqtt.username
meshtastic --set mqtt.password
```

Konfiguroitu päätelaite välittää kaiken datan *MQTT*-yhdyskäytävälle. Viestit välitetään ServiceEnvelope protokollapuskurin avulla eli varsinainen viestisisältö on paketoitu osaksi muuta dataa. ServiceEnvelope liittää dataan tietoa siitä, miltä kanavalta data on lähetetty ja mikä noodi viestin on lähettänyt. Meshtastic dokumentaatiossa suositellaan käytettäväksi Node-RED sovellusta *MQTT*-viestien käsittelyssä ja ohjeissa kuvataan mqtt.proto protokollapuskurin käyttöönotto Node-RED avulla. Node-RED käyttö on suositeltavaa siinä tapauksessa, että *MQTT*:n välittämästä datasta halutaan hyödyntää muutakin kuin tekstiviestiliikenteen sisältö joka on automaattisesti *JSON*-formaatissa.

4.6.7 Meshtastic telemetria konfiguraatio

Meshtastic verkossa voidaan tarjota yhden tai useamman päätelaitteen toimesta lämpötila, paine, kosteus tietoja. Toiminta edellyttää erillisen anturin kytkemistä päätelaitteeseen *I2C*-väylän avulla ja telemetria moduulin käyttöönottoa. *I2C* (engl.

Inter-Integrated Circuit) väylä on periaatteeltaan sarjamuotoinen ja se on tarkoitettu lyhyenmatkan viestikanavaksi oheislaitteiden välillä. Väylä on kaksisuuntainen ja oheislaitteet kytketään rinnan välylälle kahden liitynnän avulla. Liittymän kommunikaatio tapahtuu *SDA*-liitynnällä, kun taas *SCL* vastaa väylän kellosignaalista.

Yksi tuetuista anturityypeistä on *BMP280*, joka tuottaa lämpötila ja painetietoa päätelaitteen *I2C*-väylälle. Anturin kytketään päätelaitteen *SDA*- ja *SCL*-liittimiin ja näiden lisäksi tarvitaan 3.3 voltin jännite ja tähän liittyvä maadoitus eli nolla anturille. Päätelaitteen *SDA*- ja *SCL*-liitäntöjen selvittämiseen voi käyttää konfiguraatio tietoa, joka löytyy Githubista kanavalta **meshtastic/firmware/variants**. Jokaiselle päätelaitetyypille on oma hakemistonsa, joka sisältää tiedoston **variant.h**. Tiedostossa on muun muassa kuvattuna *SDA*- ja *SCL*-liittimien tiedot alla olevan esimerkin mukaisesti.

```
#define I2C_SDA 4
#define I2C_SCL 15
```

Esimerkissä numerot kuvaavat liittimen numeerista tietoa kuten *SDA* 4 ja *SCL* 15. Päätelaitteen kytkentäkaavio kuvaa liittimen fyysisen kytkentäpisteen anturille edellä kuvattujen numeroiden perusteella. Fyysisen kytkennän jälkeen päätelaitteella pitää määrittää halutut parametrit ja aktivoida moduuli komennoilla

```
meshtastic --set telemetry.device_update_interval 120
meshtastic --set telemetry.environment_screen_enabled true
meshtastic --set telemetry.environment_measurement_enabled true
```

Edellä kuvattu konfiguraatio määrittää anturiarvojen päivitysväliksi 120 sekuntia, esittää tiedon päätelaitteen näytöllä ja aktivoi moduulin. Päätelaitteen uudelleen käynnistyksen jälkeen mittaustiedot on saatavilla päätelaitteen näytöllä ja käyttöliittymäsovelluksella. Tieto välitetään myös MQTT-viestinä, mutta sen purkaminen edellyttää aiemmin mainittua Node-RED käyttöönottoa protokollapuskureiden avulla.

4.6.8 Meshtastic GPIO-konfiguraatio

Päätelaitteella olevia *GPIO*-liityntöjä (engl. General purpose input / output) voidaan ohjata sarjaporttiyhteyden avulla ja liityntöihin voidaan kohdistaa kirjoitus-, luku- ja tilaoperaatioita. Version 1.3 dokumentaatio kuvaa toiminnan niin, että vähintään kahden päätelaitteen välillä tulee muodostaa kanava nimeltä **gpio**. Tälle kanavalle voidaan kohdistaa Python komentokehotteessa komentoja, joiden avulla voidaan manipuloida tai tarkastella päätelaitteella olevan *gpio*-liittymää. Komennot

muodostuvat lähde, toiminto, kohde logiikalla, missä lähdettä edustaa paikallinen tietokone parametrin *port* avulla. Port kuvaa sarjaportin missä komentoa suoritetaan. Toimintoa edustaa *gpio-rd*, *gpio-wrb* tai *gpio-watch* opeaattorit ja *dest* operaattorin avulla toiminto kohdistetaan tietylle päätelaitteelle. *Gpio-wrb* operaattori mahdollistaa portin tilan muuttamisen 0- ja 1-arvojen välillä, eli komennon formaatti on *gpio-wrb* liittymänumero ja tilanumero.

```
meshtastic --port /dev/ttyUSB0 --gpio-wrb 2 0 --dest \!f244c480
```

gpio-rd operaattorilla voidaan pyytää *GPIO*-liitynnän tilatietoa. Kysymys tulee esittää hexadesimaalina joka perustuu portin numeroon. Numero voidaan laskea korottamalla numero 2 liittymänumeron potenssiin ja ilmaisemalla arvo hexadesimaalina.

$$Hex = 2^{gpio}$$

Jos tarkastellaan liittymänumeroa 4. $2^4 = 0x10$

```
meshtastic --port /dev/ttyUSB0 --gpio-rd 0x10 --dest \!f244c480
```

gpio-watch toiminta poikkeaa edellä kuvatusta *gpio-rd* operaattorista sillä, että kysymys jää luuppiin. Kyselyn voi keskeyttää cltr-c komennolla.

```
meshtastic --port /dev/ttyUSB0 --gpio-watch 0x10 --dest \!f244c480
```

4.7 Sovellus

Kuvassa 4.1 on esitelty *MQTT*-tilaajana toimiva sovellus, jonka tehtävänä on tallentaa *MQTT*-yhdyskäytävältä tilatun *JSON*-viestiliikenteen sisältö tietokantaan. Alaluvussa 4.6.6 on kuvattu päätelaitteen käyttöönotto ja mainittu suositus Node-RED käytöstä. Jos sovellus luodaan ilman Node-RED:in *MQTT*-välittäjäpalvelua, esiintyy viestisisältö tilaajalla ohjelmakoodi 4.2 mukaisesti.

Ohjelmakoodi 4.2: Esimerkki MQTT-viestin sisällöstä

```
@ğ"
5CHX
testikanava      !8487d440
{"channel": 0, "from": -2071473088, "id": 2135154566, "payload":
{"air_util_tx": 0, "battery_level": 0, "channel_utilization": 0,
"voltage": 0}, "sender": "!8487d440", "timestamp": 0, "to": -1,
"type": "telemetry"}
```

Meshtastic *MQTT* Node-RED ohjeissa kuvataan *JSON*-datan purku perustuen protokollapuskureiden käyttöön. Koska *MQTT*-yhdyskäytävä ei ole ole luotu Node-RED alustalla esiintyy *JSON*-sanomilla purkamatonta dataa ohjelmakoodi 4.2 mukaisesti.

4.7.1 JSON funktio

JSON-aineoston käsittely ratkaistiin sovelluksella niin, että jokainen *MQTT*-välittämä viesti annetaan syötteenä funktiolle. Funktio tarkistaa, onko syöte *JSON*-muotoista. Ohjelmakoodi 4.3 saa syötteenä *MQTT*-sanoman ja palauttaa boolean arvon onko syöte *JSON*-formaatissa. Syötteen tallentamien tietokantaan edellyttää, että data on JSON-formaatissa

Ohjelmakoodi 4.3: JSON-funktio

```
1 import json
2
3 def is_json(checkdata):
4   try:
5     json.loads(checkdata)
6   except ValueError as e:
7     return False
8   return True
```

4.7.2 SQL-tietokanta ja taulunmuodostus

Tietokantana käytetään Pythonin mukana tulevaa sqlite-tietokantaa. Toteutuksessa olisi voinut olla käytössä mikä tahansa tietokanta, mutta yksinkertaisuuden nimissä teidontallennus esimerkki on toteutettu tämän avulla. Ohjelmakoodissa 4.4 riveillä 4-10 on funktio, jonka avulla muodostetaan tietokanta. Tietokannan muodostus saa syötteenä rivin 20-21 parametrit, missä kuvataan tietokantatiedoston sijainti ja muodostetaan tietokantaobjekti myöhempää käyttöä varten. Riveillä 12 - 17 on funktio, joka muodostaa taulun mikäli sitä ei ole olemassa, tietokantaobjektin ja riveillä 22-25 olevan määrityksen mukaisesti. Ohjelmakoodia 4.4 pitää kutsua kerran ennen pääsovelluksen käynnistämistä.

Ohjelmakoodi 4.4: Sqlite. Tietokannan ja taulun muodostus

```
1 import sqlite3
2 from sqlite3 import Error
3
```

```
4  def create_connection(db_file):
5      conn = None
6      try:
7          conn = sqlite3.connect(db_file)
8      except Error as e:
9          print(e)
10     return conn
11
12 def create_table(conn, create_table_sql):
13     try:
14         c = conn.cursor()
15         c.execute(create_table_sql)
16     except Error as e:
17         print(e)
18
19 def generate_table():
20     database = r"C:\dev\meshtastics\mqttclient\pythonsqlite.db"
21     conn = create_connection(database)
22     sql_create_mqtt_table = """ CREATE TABLE IF NOT EXISTS MQTT (
23                                     id integer AUTOINCREMENT,
24                                     textdata text,
25                                     insertdate text
26                                 );"""
27     if conn is not None:
28         create_table(conn, sql_create_mqtt_table)
29     else:
30         print("Error! cannot create the database connection.")
31
32 if __name__ == '__main__':
33     generate_table()
```

Windows ympäristössä tietokannan tauluja ja siellä olevaa dataa voi tarkastella erikseen asennettavalla sqlite3.exe ohjelmistolla. Asennuksen jälkeen tietokantatiedostoa voidaan kutsua komennolla sqlite3.exe tietokantatiedoston nimi. Em. komento avaa komentokehotetulkin tietokantaan. Muodostetut taulut saa esiin komennolla

```
.tables
```

ja mikäli halutaan tarkastella taulujen sarakenimiä tapahtuu se komennolla.

```
PRAGMA table_info(table_name);
```

4.7.3 Tiedon tallennus SQL-tietokantaan

Meshtastic kanavalta saapunut data tallennetaan ohjelmakoodissa 4.5 kuvattujen funktioiden avulla ohjelmakoodissa 4.4 luotuun tietokantaan. Riveillä 20-25 on kuvattu Add_data_db() fuktio, joka saa syötteenä MQTT-tilaajalta datan, jonka se tallentaa tietokantaan riveillä 12-18 esitellyn from_mqtt() funktion avulla. Riveillä 4-10 esitelty create_connection() funktio tuottaa objektin, minkä avulla varsinainen tietokantayhteys muodostetaan.

Ohjelmakoodi 4.5: Tiedon tallennus SQL-tietokantaan

```
1  import sqlite3
2  from sqlite3 import Error
3
4  def create_connection(db_file):
5      conn = None
6      try:
7          conn = sqlite3.connect(db_file)
8      except Error as e:
9          print(e)
10     return conn
11
12 def from_mqtt(conn, task):
13     sql = ''' INSERT INTO MQTT(textdata,insertdatetime)
14               VALUES(?,?) '''
15     cur = conn.cursor()
16     cur.execute(sql, task)
17     conn.commit()
18     return
19
20 def add_data_db(mqttdata):
21     database = r"C:\dev\meshtastics\mqttclient\pythonsqlite.db"
22     conn = create_connection(database)
23     with conn:
24         task = (mqttdata['channeldata'], mqttdata['datetimestr'])
25         from_mqtt(conn, task)
```

4.7.4 Sovelluksen main funktio

Ohjelmakoodissa 4.7 on kuvattu *MQTT*-tilaaja funktio, jota suoritetan loputtomassa luupissa (ohjelmakoodin rivi 45) . Funktio tilaa MQTT-yhdyskäytävältä dataa ja tallentaa sen soveltuvasti tietokantaan. Tilauksen konfiguraatio on kuvattu ohjelma-

koodin 4.7 riveillä 6 - 11. *MQTT*:n käyttäminen edellyttää *Paho*-kirjaston asentamista, joka mahdollistaa *MQTT*-toiminteet sovelluksella. Kirjasto asennetaan komennolla.

```
pip install paho-mqtt
```

Tilatun JSON-sanoman sisältö luokittuu eri tyyppeihin objektin avaimen *type* perusteella. Yksi tyypeistä on *text* ohjelmakoodi 4.6 esimerkin mukaisesti. Vastaava esimerkki löytyy ohjelmakoodin vastauksesta 4.2 avaimella *telemetry*.

Ohjelmakoodi 4.6: Esimerkki MQTT-tilaaja viestin sisällöstä

```
{"channel": 0, "from": -2069944272, "id": -1994245121,
"payload": {"text": "Meshtastic kanavalla välitetty tekstiviesti"},
"sender": "!8487d440", "timestamp": 0, "to": -1, "type": "text"}
```

Meshtastic kanavalta tulevassa datassa (ohjelmakoodi 4.6) on avain *timestamp*, joka esiintyy arvoilla nolla. Mitä ilmeisemmin kyseessä on Meshtastic sovellusversion virhe ja tässä pitäisi esiintyä aikaleima. Tietokantaan tallentavalle datalle haluttiin muodostaa aikaleima, joka esiintyy ohjelmakoodin 4.7 riveillä 27-28. Toinen tietokantaan tallennettava tieto on *text* avaimen arvo. Sisällöllisesti tämä on se mitä käyttöliittymällä on kirjoitettu kanavalle tekstiviestinä. Tallennettava arvo saadaan *JSON*-objektilta viittaamalla avaimeen objektin *mqttdata['payload']['text']* osaan. Ohjelmakoodin 4.7 riveillä 31 - 35 muodostetaan Python sanakirja objekti tekstiviestin sisällöstä ja aikaleimasta ja tämä välitetään funktiolle joka tallentaa datan kantaan. Ohjelmakoodin 4.7 riveillä 13 - 23 on *MQTT*-funktio, joka muodostaa yhteyden *MQTT*-yhdyskäytävälle. Riveillä 25 - 37 on *MQTT*-tilaaja joka toimii samalla myös datan tallentajana tietokantaan. Rivillä 29 tarkistetaan onko saapunut viesti sisällöllisesti *JSON*-formaatissa. Rivillä 31 tarkisteaan sisältääkö JSON-obkekti avaimen *text*.

MQTT-yhteys funktion alkuperäinen kirjoittaja on Dekum Tao ja sitä on muokattu tarkoitusperään soveltuvasti.

Ohjelmakoodi 4.7: Main funktio

```
1 from datetime import datetime
2 import json
3 from paho.mqtt import client as mqtt_client
4 import sql_db as todb
5
6 broker = 'x.x.x.x'
7 port = 1883
```

```
 8 topic = "msh/#"
 9 client_id = f'yhdyskaytava_sovellus'
10 username = ''
11 password = ''
12
13 def connect_mqtt() -> mqtt_client:
14     def on_connect(client, userdata, flags, rc):
15         if rc == 0:
16             print("Connected to MQTT Broker!")
17         else:
18             print("Failed to connect, return code %d\n", rc)
19     client = mqtt_client.Client(client_id)
20     client.username_pw_set(username, password)
21     client.on_connect = on_connect
22     client.connect(broker, port)
23     return client
24
25 def subscribe(client: mqtt_client):
26     def on_message(client, userdata, msg):
27         now = datetime.now()
28         datetime_string = now.strftime("%d-%m-%Y %H:%M:%S")
29         if is_json(msg.payload.decode('utf-8', 'ignore')):
30             mqttdata = json.loads(msg.payload.decode('utf-8', 'ignore'))
31             if mqttdata['type'] == 'text':
32                 parsedmqttdata = dict({"channeldata" :
33                 mqttdata['payload']['text'],
34                 "datetimestr": datetime_string})
35                 todb.add_data_db(parsedmqttdata)
36     client.subscribe(topic)
37     client.on_message = on_message
38
39 def run():
40     client = connect_mqtt()
41     subscribe(client)
42     client.loop_forever()
43
44 if __name__ == '__main__':
45     run()
```

4.8 Yhteenveto käytännön toteutuksesta

Käytännön toteutus lähti liikkeelle soveltuvien päätelaitteiden tilaamisesta ja valinta kohdistui halvimpaan Heltec päätelaitemalliin, missä ei ole GPS-tukea. Laitteet tilattiin suoraan Kiinalaiselta valmistajalta ja toimitusaika oli noin kolme viikkoa. Päätelaitetta voisi myös simuloida Linux virtuaalikoneena, mutta ei olisi ollut kestävä ratkaisu, koska tarkastelussa on laitteen ulkopuolisia yhteyksiä esimerkiksi mittalaitteiden osalta.

Sovelluskehitys ja konfiguraatiot aloitettiin Mehstastic versiolla 1.2. Päätelaitteiden käyttöönotto on selkeä toimenpide puhuttaessa pelkästään tekstiviestin lähettämisestä kahden puhelimen tai tabletin välillä. Riittää, että käytössä on yhteensopivat, eli saman versioiset käyttöliittymä- ja laiteohjelmistoversiot. Riippuen käytettävistä päätelaitemerkistä, laiteohjelmistoversioiden päivittäminen voi olla hankalaa. Ohjelmistot millä päivityksiä tehdään eivät välttämättä tunnista päätelaitetta tai sen versiota ja samasta päätelaitetyypistä voi olla markkinoilla monia eri variantteja. Päivityksen yhteydessä tulee kiinnittää erityistä huomiota siihen, että laiteohjelmisto (engl. firmware) vastaa päätelaitteen laitteisto (engl. hardware) versiota.

Ensimmäiset WiFi, telemetria ja MQTT-testit tehtiin perustuen version 1.2 ohjeistukseen ja melko nopeasti selvisi, että tekstiviestien sisältö on mahdollista purkaa MQTT-sanoman sisältä melko yksinkertaisesti, mutta telemetriatieto esiintyy siellä merkistönä mistä ei saanut selvää. Meshtastic tukifoorumi suositteli luopumaan Heltechin päätelaitteiden käytöstä MQTT-käyttötapauksissa vajavaisen MQTT-JSON sanomamuodostuksen vuoksi. Tämän takia tilasin LiLyGo T-Beam päätelaitteita ja näiden saavuttua siirryttiin Meshtastic versioon 1.3. Pävitetyissä ohjeissa esiteltiin Node-RED perusteinen MQTT-yhdyskäytävä, joka tukee protokolla puskureiden avulla tapahtuvaa telemetria tiedon purkua MQTT-viestiliikenteestä. Toisin sanoen alkuperäinen ongelma väärän päätelaitetyypin käytöstä ei pitänyt paikkaansa. Mosquitto perusteinen MQTT-yhdyskäytävä oli jo tässä vaiheessa toteutettu ja ajankäytön kannalta en ryhtynyt ottamaan Node-Red perusteista ratkaisua käyttöön. MQTT sanomassa oleva JSON-purku käyttäytyi samoin Heltech:in ja LiLyGo päätelaitteilla. Ilman Node-Red MQTT-palvelua telemetriatiedon purkamien MQTT-viestistä ei ole mahdollista ja tämä selvisi hyvinkin myöhäisessä vaiheessa.

Version 2.0 tullessa ulos Heltec päätelaitteiden tuki loppui virallisesti kokonaan. Näillekin päätelaitteille on kuitenkin saatavilla laiteohjelmistoja, mutta ei suositusta näiden päätelaitteiden käytölle.

Dokumentin kirjoitushetkellä heinäkuu 2022 - joulukuu 2022 Meshtastic on jul-

kaissut uusia pääsovellusversioita 3 kappaletta. Alpha- ja Beta-versioita julkaistaan viikoittain ja kehittäjän voi olla vaikea pysyä mukana, miten tuote kehittyy minkäkin julkaisun sisällä. Positiivista on se, että sovellus kehittyy nopealla tahdilla. Esimerkiksi 1.2 versiossa laiteohjelmistopäivitykset tehtiin ainoastaan komentokehotteesta. 1.3 versiossa esiteltiin graaffinen käyttöliittymä päivityksille ja versiossa 2.0 laiteohjelmistopäivitykset on mahdollista tehdä myös web-selaimen avulla.

Koska kyseessä on yhteisönkehittämä sovellus, ohjeiden- ja ohjelmistoversioiden yhteensovittamisessa on myös haasteita. Web-sivustolla olevat ohjeet voivat muuttua yhden sivupäivityksen välillä ja aiemmin julkaisut ohjeet häviävät ja niihin ei enää myöhemmin ole pääsyä. Tukifoorumi toimii kuten mikä tahansa julkinen tukifoorumi. Vastaukset voivat olla hyvinkin tarkkoja ja joissain tapauksissa kysymyksistä saattaa muodostua kehitystikettejä, mikäli kyseessä on ohjelmisto virhe. Vastaavasti ratkaisuehdotukset voivat olla harhaanjohtavia kuten MQTT-telemetria tapauksessa kävi. Olennaisinta on kuitenkin se, että tukea on saatavilla ja sitä pitää osata hyödyntää oikealla tavalla.

5 Yhteenveto

Pro gradussa esitellään *LoRa*:n ja *LoRaWAN*:in toimintaperiaatteita ja näihin liittyvä tutkimustuloksia. Tarkastelussa on myös *LoRa*:n perustuva Meshtastic sovelluskehikko, jonka avulla muodostetaan esimerkkisovellus telemetria ja tiedonsiirtoverkosta.

Kappale 2 tarjoaa perusteet *LoRa*:n ja *LoRaWAN*:in osalta. Samassa kappaleessa oleva *LoRaWAN* osuus esittelee lukijalle laajasti käytössä olevan toteutustavan, minkä varjopuolina on melko monimutkainen infrastruktuurirakenne ympäristörajoitteineen. Langattomista IoT-tiedonsiirtotekniikoista *LoRAWAN* tarjoaa erinomaisen *LPWAN* mukaisen toimintaympäristön päätelaitehallinnan, virrankulutuksen ja tietoturvan osalta varasinkin kun käytössä on A-luokassa toimiva päätelaite.

Kappaleessa 2 kuvatun *LoRa*:n osalta voidaan todeta, että hajaspektrimoduloinnin tarjoama erinomainen kuuluvuus ja häiriönsietokyky vähäisellä lähetysteholla kiinnostaa niin tutkijoita kuten myös sovelluskehittäjiäkin. *LoRa*:n haittapuoleksi voidaan nostaa sen käyttämä ISM-taajuusalue ja sen määrittämät liikennöintimäärärajoitteet. Liikennöintirajoitteiden noudattaminen on pitkälti sovelluksenkehittäjän vastuulla ja valvonta voi olla hyvinkin vaikea toteuttaa käytännön tasolla. *LoRa*:n haitaksi voidaan myös katsoa vapaasti saatavilla olevien sovelluskirjastojen melko rajallinen saatavuus.

LoRa:n käyttöoikeuskysymykset ja tähän liittyvä lisensointi eivät ole yksiselitteisiä. *LoRa*-allianssi toimii etujärjestönä *LoRaWAN*-tekniikan edistämiselle ja tätä kautta eri valmistajia rohkaistaan valmistamaan esimerkiksi *LoRa*-radio piirejä. Tässä huomio siihen, että *LoRa*:n omistaa Semtech, joka on em. allianssin jäsen ja on luovuttanut *LoRa*:n käyttöoikeuden allianssille. Kuten pro gradun käytännön osuudessa on kuvattu *LoRa*-radioita voidaan käyttää ilman *LoRaWAN* kytkentää ja voidaankin kysyä, että onko tämä omiaan edistämän *LoRa*-allianssin tavoitteita? *LoRAWAN*-tekniikkaan käyttöoikeuden voi lisensoida ja lisenssimaksu kattaa molempien tekniikoiden käyttöoikeuden. *LoRa*:a ei ole tarkoitettu erikseen lisensoitavaksi. Edellä olevaa ei pidä tulkita niin, etteikö pelkästään *LoRa*:n perustuvia ratkaisuja kannatta kehittää, mutta lisensoitiin liittyvät kysymykset on syytä selvittää ennen kehitetyn tuotteen kaupallistamista.

Pro gradun kappaleessa 3 tarkastellaan välittäviä ja reitittäviä *LoRa*-modulointiin perustuvia mesh-toteutuksia. Esitellyt ratkaisuedotukset ovat hyvin erityyppisiä ja niissä sovelletaan olemassa olevien protokollien toimintatapoja. Viimekädessä kehitettävä sovellus määrittää miten se kannttaa teknisesti toteuttaa. Sovelluskehystä valittaessa on huomioitavia, miten sen avulla voidaan vastata seuraaviin kysymyksiin. Miten ylläpidettävyys, virrankulutus, tietoturva, kokonaiskustannukset ja toimintavarmuus on huomioitu kehyksen toiminnassa.

Kappaleessa 4 tarkastellaan Meshtastic sovelluskehystä. Kysymykseen voidaanko Meshtastic:n avulla tuottaa viesti ja mittausverkko palvelut samanaikaisesti, vastaus on kyllä. Meshtastic tarjoaa avoimen sovelluskehikon, jonka avulla on mahdollista luoda omia sovelluksia hyväksikäyttäen *LoRa*-mesh mallia. Tekniikkaa voi hyödyntää niin tekstiviesti kuten automaatio tai mittalaitekäyttötapauksissa ja se soveltuu käytettäväksi sellaisenaan esimerkiksi katastrofialueilla. Tekniikan haittapuolena voidaan pitää verkon kuormitusta tulvimiseen perustuvan liikennöintitavan takia ja tähän liittyvää päätelaitteiden virrankulutusta.

Meshtastic projektissa on suuri potentiaali. Se tarjoaa yksinkertaisen, melko helposti lähestyttävän kokonaisuuden, missä kehittäjällä on mahdollisuus käyttää sovellusta sellaisenaan tai muokata kehikkoa omien tarpeidensa mukaisesti. Meshtastic:n avulla on helppo luoda toteutus, missä päätelaitteet tuottavat ja tai vastaanottavat dataa. Jokainen päätelaite voi toimia yhdyskäytävänä, reitittimenä, GPIO-alustana tai vain päätelaitteena, joka mahdollistaa viestin lähettämisen ja vastaanottamisen. GPIO:n osalta voidaan todeta, että perusohjelmistolla päätelaite voi toimia mittalaitteena tai sen avulla voidaan ohjata vaikkapa etäältä relettä. Jokaisen päätelaitteen lisääminen kasvattaa verkon peittoaluetta tiettyyn rajaan asti. Loppukäyttäjän, jolla ei ole kiinnostusta tehdä sovelluskehitystä projekti tarjoaa halvan ja yksinkertaisen tekstiviestikommunikointi alustan, mitä voi käyttää missä tahansa riippumatta siitä, onko tietoliikenneyhteydet saatavilla.

Meshtastic projektilla on rajattomasti sovelluskohteita ja kuten markkinataloudessa on tapana toimia en pitäisi mahdottomana, vaikka projektin lopputulos myytäisiin ja tai muuten kaupallistettaisiin myöhemmässä vaiheessa.

Lähteet

[1] ABRARDO, A., JA POZZEBON, A. A multi-hop LoRa linear sensor network for the monitoring of underground environments: the case of the Medieval Aqueducts in Siena, Italy. *Sensors 19*, 2 (2019), 402.

[2] ASLAM, M. S., KHAN, A., ATIF, A., HASSAN, S. A., MAHMOOD, A., QURESHI, H. K., JA GIDLUND, M. Exploring multi-hop lora for green smart cities. *IEEE Network 34*, 2 (2019), 225–231.

[3] BOR, M., VIDLER, J. E., JA ROEDIG, U. LoRa for the Internet of Things.

[4] CENTELLES, R. P., FREITAG, F., MESEGUER, R., JA NAVARRO, L. Beyond the star of stars: An introduction to multihop and mesh for LoRa and LoRaWAN. *IEEE Pervasive Computing 20*, 2 (2021), 63–72.

[5] CHOI, R., LEE, S., JA LEE, S. Reliability improvement of lora with arq and relay node. *Symmetry 12*, 4 (2020), 552.

[6] COTRIM, J. R., JA KLEINSCHMIDT, J. H. LoRaWAN Mesh Networks: A Review and Classification of Multihop Communication. *Sensors 20*, 15 (2020).

[7] DIAS, J., JA GRILO, A. LoRaWAN multi-hop uplink extension. *Procedia computer science 130* (2018), 424–431.

[8] DIAS, J., JA GRILO, A. Multi-hop LoRaWAN uplink extension: specification and prototype implementation. *Journal of Ambient Intelligence and Humanized Computing 11*, 3 (2020), 945–959.

[9] DUONG, C. T., JA KIM, M.-K. Reliable multi-hop linear network based on LoRa. *Int. J. Control Autom 11*, 4 (2018), 143–154.

[10] DWIJAKSARA, M. H., JEON, W. S., JA JEONG, D. G. Multihop gateway-to-gateway communication protocol for lora networks. Julkaisusarjassa *2019 IEEE International Conference on Industrial Technology (ICIT)* (2019), IEEE, 949–954.

[11] EBI, C., SCHALTEGGER, F., RÜST, A., JA BLUMENSAAT, F. Synchronous LoRa Mesh Network to Monitor Processes in Underground Infrastructure. *IEEE Access 7* (2019), 57663–57677.

[12] ECMA INTERNATIONAL. ECMA-404. The JSON data interchange syntax 2nd edition, December 2017. URL https://www.ecma-international.org/publications-and-standards/standards/ecma-404/, viitattu 8.11.2022.

[13] ERTÜRK, M. A., AYDN, M. A., BÜYÜKAKKALAR, M. T., JA EVIRGEN, H. A Survey on LoRaWAN Architecture, Protocol and Technologies. *Future Internet 11*, 10 (2019).

[14] GOOGLE. Protocl Buffers. URL https://developers.google.com/protocol-buffers, viitattu 2.10.2022.

[15] HAUBRO, M., ORFANIDIS, C., OIKONOMOU, G., JA FAFOUTIS, X. TSCH-over-LoRA: long range and reliable IPv6 multi-hop networks for the internet of things. *Internet Technology Letters 3*, 4 (2020), e165.

[16] HILLAR, G. C. *MQTT Essentials-A lightweight IoT protocol*. Packt Publishing Ltd, 2017.

[17] HUSSAIN, M. I., AHMED, N., AHMED, M. Z. I., JA SARMA, N. QoS provisioning in wireless mesh networks: A survey. *Wireless Personal Communications 122*, 1 (2022), 157–195.

[18] ISO/IEC 20922:2016. Message Queuing Telemetry Transport (MQTT) v3.1.1. URL https://www.iso.org/obp/ui/#iso:std:iso-iec:20922:ed-1:v1:en, viitattu 25.10.2022.

[19] K. HESTER ET AL., MESHTASTIC. An opensource hiking, pilot, skiing, secure GPS mesh communicator, 2020. URL https://www.meshtastic.org/, viitattu 10.9.2022.

[20] KREJÍ, R., HUJÁK, O., JA VEPE, M. Security survey of the IoT wireless protocols. Julkaisusarjassa *2017 25th Telecommunication Forum (TELFOR)* (2017), 1–4.

[21] LAYA, A., KALALAS, C., VAZQUEZ-GALLEGO, F., ALONSO, L., JA ALONSO-ZARATE, J. Goodbye, aloha! *IEEE access 4* (2016), 2029–2044.

[22] LIAO, C.-H., ZHU, G., KUWABARA, D., SUZUKI, M., JA MORIKAWA, H. Multi-Hop LoRa Networks Enabled by Concurrent Transmission. *IEEE Access 5* (2017), 21430–21446.

[23] LONGMAN, E., EL-HAJJAR, M., JA MERRETT, G. V. Multihop Networking for Intermittent Devices. Julkaisusarjassa *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems* (2022), 878–884.

[24] LORA ALLIANCE. About Lora Alliance. URL `https://lora-alliance.org/about-lora-alliance/`, viitattu 10.9.2022.

[25] LORA ALLIANCE. LoRaWAN 1.0.3 Specification. URL `https://lora-alliance.org/wp-content/uploads/2020/11/lorawan1.0.3.pdf`, viitattu 10.8.2022.

[26] LORA ALLIANCE. RP002-1.0.1 LoRaWAN Regional Parameters. URL `https://lora-alliance.org/wp-content/uploads/2020/11/rp_2-1.0.1.pdf`, viitattu 10.8.2022.

[27] LORA ALLIANCE. TS001-1.0.4 LoRaWAN L2 1.0.4 Specification. URL `https://lora-alliance.org/wp-content/uploads/2021/11/LoRaWAN-Link-Layer-Specification-v1.0.4.pdf`, viitattu 10.8.2022.

[28] LUNDELL, D., HEDBERG, A., NYBERG, C., JA FITZGERALD, E. A routing protocol for LoRa mesh networks. Julkaisusarjassa *2018 IEEE 19th International Symposium on"A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)* (2018), IEEE, 14–19.

[29] M. MCCAULEY. RadioHead Packet Radio Library for embedded microprocessors, 2014. URL `https://www.airspayce.com/mikem/arduino/RadioHead/`, viitattu 10.9.2022.

[30] MAI, D. L., JA KIM, M. K. Multi-hop LoRa network protocol with minimized latency. *Energies 13*, 6 (2020), 1368.

[31] MAMOUR, D., JA CONGDUC, P. Increased flexibility in long-range IoT deployments with transparent and light-weight 2-hop LoRa approach. Julkaisusarjassa *2019 Wireless Days (WD)* (2019), IEEE, 1–6.

[32] NOREEN, U., BOUNCEUR, A., JA CLAVIER, L. A study of LoRa low power and wide area network technology. Julkaisusarjassa *2017 International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)* (2017), 1–6.

[33] PYCOM, PYMESH. LoRa full-mesh network technology., 2020. URL `https://docs.pycom.io/pymesh/`, viitattu 10.9.2022.

[34] SAELENS, M., HOEBEKE, J., SHAHID, A., JA POORTER, E. D. Impact of EU duty cycle and transmission power limitations for sub-GHz LPWAN SRDs: An overview and future challenges. *EURASIP Journal on Wireless Communications and Networking 2019*, 1 (2019), 1–32.

[35] SARTORI, B., THIELEMANS, S., BEZUNARTEA, M., BRAEKEN, A., JA STEENHAUT, K. Enabling RPL multihop communications based on LoRa. Julkaisusarjassa *2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)* (2017), 1–8.

[36] SEMTECH. Lora network packet forwarder project. URL `https://github.com/lora-net/packet_forwarder`, viitattu 15.5.2023.

[37] SEMTECH, A., JA BASICS, M. AN1200. 22. *LoRa Modulation Basics 46* (2015).

[38] SEMTECH CORPORATION. What are LoRa and LoRaWAN? URL `https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan/`, viitattu 10.8.2022.

[39] SHANMUGA SUNDARAM, J. P., DU, W., JA ZHAO, Z. A Survey on LoRa Networking: Research Problems, Current Solutions, and Open Issues. *IEEE Communications Surveys Tutorials 22*, 1 (2020), 371–388.

[40] SISINNI, E., CARVALHO, D. F., FERRARI, P., FLAMMINI, A., SILVA, D. R. C., JA DA SILVA, I. M. Enhanced flexible LoRaWAN node for industrial IoT. Julkaisusarjassa *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)* (2018), IEEE, 1–4.

[41] SISINNI, E., FERRARI, P., CARVALHO, D. F., RINALDI, S., MARCO, P., FLAMMINI, A., JA DEPARI, A. LoRaWAN range extender for Industrial IoT. *IEEE Transactions on Industrial Informatics 16*, 8 (2019), 5607–5616.

[42] SPRINGER, A., GUGLER, W., HUEMER, M., REINDL, L., RUPPEL, C., JA WEIGEL, R. Spread spectrum communications using chirp signals. Julkaisusarjassa *IEEE/AFCEA EUROCOMM 2000. Information Systems for Enhanced Public Safety and Security (Cat. No. 00EX405)* (2000), IEEE, 166–170.

[43] TANJUNG, D., BYEON, S., KIM, D. H., JA KIM, J. D. Oodc: An opportunistic and on-demand forwarding mechanism for lpwa networks. Julkaisusarjassa *2020 International Conference on Information Networking (ICOIN)* (2020), IEEE, 301–306.

[44] THETHINGSNETWORK. LoRaWANő distance world record broken, 2019. URL `https://www.thethingsnetwork.org/article/lorawan-distance-world-record`, viitattu 25.11.2022.

[45] YI, J., CLAUSEN, T., JA TOWNSLEY, W. C. Systems,A study of LoRa: long range & low power networks for the Internet of Things,. *Sensors 16*, 9 (2016).

[46] ZHOU, W., TONG, Z., DONG, Z. Y., JA WANG, Y. LoRa-Hybrid: A LoRaWAN Based Multihop Solution for Regional Microgrid. Julkaisusarjassa *2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)* (2019), 650–654.

[47] ZHU, G., LIAO, C.-H., SAKDEJAYONT, T., LAI, I.-W., NARUSUE, Y., JA MORIKAWA, H. Improving the Capacity of a Mesh LoRa Network by Spreading-Factor-Based Network Clustering. *IEEE Access 7* (2019), 21584–21596.