

Mesh-verkkotopologian toteuttaminen LoRa-tekniikalla

Miika Avela

Opinnäytetyö

Joulukuu 2019

Tekniikan ja liikenteen ala

Insinööri (AMK), ohjelmistotekniikan tutkinto-ohjelma

Tekijä(t) Avela, Miika	Julkaisun laji Opinnäytetyö, AMK	Päivämäärä Joulukuu 2019
	Sivumäärä 78	Julkaisun kieli Suomi
		Verkojulkaisulupa myönnetty: x
Työn nimi Mesh-verkkotopologian toteuttaminen LoRa-tekniikalla		
Tutkinto-ohjelma Ohjelmistotekniikan tutkinto-ohjelma		
Työn ohjaaja(t) Marko Rintamäki, Esa Salmikangas		
Toimeksiantaja(t)		
<p>Tiivistelmä</p> <p>Jatkuvasti kasvava verkon käyttöaste luo uusia haasteita nykyisille tiedonsiirtoverkoille ja tietoliikenneprotokollille. Internet-verkkoon nykyisin kohdistuvan kuorman määrää voitaisiin keventää siirtymällä Internet of Things-järjestelmissä vaihtoehtoisin, vähävirtaisiin tiedonsiirtoratkaisuihin. LoRa-tekniikalla toteutetun tiedonsiirtoverkon ja sitä tukevan tietoliikenneprotokollan soveltuvuutta tutkittiin vaihtoehtoisena tiedonsiirtoratkaisuna perinteisen internet-verkon tilalle.</p> <p>Yleisimpien tietoliikenneprotokollien pätevyyttä vaaditun verkon toteutuksessa tarkasteltiin ja niiden pohjalta sovellettiin LoRa-tekniikkaan yhteensopiva tietoliikenneprotokolla, joka mahdollisti verkon solmulaitteiden välisen tietoliikenteen monivaiheisesti ja ryhmäkohtaisesti.</p> <p>Kehittämistutkimuksen tulokset saatiin aineistolähtöisen sisällönanalyysin ja induktiivisen päättelyketjun pohjalta. Tulosten perusteella syntyi soveltuvuustutkimus prototyyppilaitteesta, jolla kehittämistutkimuksen tuloksia kyettiin teorian lisäksi arvioimaan myös käytännössä ja tunnistamaan uuden tietoliikenneprotokollan kehityskohteet.</p> <p>Kehittämis- ja soveltuvuustutkimusten perusteella LoRa-tekniikalla voidaan luoda luotettava ja vaatimukset täyttävä tiedonsiirtoverkko. Maksimaalista kantamaa tavoitellessa pakettien lähetysajat venyvät tosin niin pitkiksi, että solmulaitteiden ja kuormituksen lisääminen hidastaa yksittäisten solmulaitteiden tiedonsiirtokykyä lineaarisesti. Tähän vaikuttavat luvasta vapaiden radiolähettimien käytölle asetetut määräykset, joiden keventäminen nostaisi verkon kuormituskykyä. Verkon solmulaitteet tulisi myös konfiguroida käyttötapauskohtaisesti vaaditulle lähetysetaisyydelle.</p>		
Avainsanat (asiasanat)		
LoRa, Mesh, Tietoliikenneprotokolla		
Muut tiedot (Salassa pidettävät liitteet)		

Author(s) Avela, Miika	Type of publication Bachelor's thesis	Date December 2019 Language of publication: Finnish
	Number of pages 78	Permission for web publication: x
Title of publication Implementing mesh network topology using LoRa-technology		
Degree programme Software Engineering		
Supervisor(s) Marko Rintamäki, Esa Salmikangas		
Assigned by		
Abstract <p>Constantly growing network utilization introduces new challenges for existing data transfer networks and communication protocols. The load towards internet network could be relieved by switching Internet of Things systems over to distinct low-powered data transfer solutions. The suitability of a LoRa communication network and its supporting communication protocol was investigated as an alternative communication solution to the traditional internet network.</p> <p>The prowess of the most common communication protocols for the implementation of the required network was examined in a development study and based on this, a communication protocol compatible with the LoRa technology was applied, which enabled for multistage and multicast communication between network nodes.</p> <p>The results of the development study were based on data-driven content analysis and inductive chain of reasoning. Based on the results, a feasibility study on a prototype device was created, which in addition to theory allowed to evaluate the results in practice and identified development areas of the new communication protocol.</p> <p>Based on the development and feasibility studies, it is attainable to create a reliable and compliant communication network using LoRa technology. However, while in pursuit for the maximum reach, packet delivery times increase so remarkably that increasing the count of node devices and the amount of load in the network slows down the data transfer rate of an individual node device in a linear fashion. Regulations set for permission free radio transmitters highly affect this and relieving such regulations would allow for increased network load capability. Furthermore, all network nodes should be configured as per use case aiming for required transmission distance.</p>		
Keywords/tags (subjects) LoRa, Mesh, Network protocol		
Miscellaneous (Confidential information)		

Sisältö

Käsitteet	5
1 Johdanto	7
2 Työn tavoitteet ja tutkimusongelma	8
2.1 Tutkimustavoitteet	8
2.2 Tutkimuskysymykset	8
2.3 Toteutetut käyttöskenaariot	9
2.3.1 Skenaario A: Viestin suora lähetys laitteelta toiselle	9
2.3.2 Skenaario B: Viestin välillinen lähetys laitteelta toiselle	10
3 Langaton tiedonsiirto ja verkkoprotokollat	11
3.1 Yleistä	11
3.2 Langaton tiedonsiirto	13
3.3 LoRa	15
3.3.1 Perusteet	15
3.3.2 Chirp.....	16
3.3.3 Spreading Factor	17
3.3.4 Coding Rate (CR)	18
3.3.5 Kaistanleveys	19
3.4 Verkon rakenne	22
3.5 Tunnetut protokollat	25
3.5.1 Protokollien valinta.....	25
3.5.2 TCP	25
3.5.3 UDP	28
4 Opinnäytetyön verkkoprotokolla	29
4.1 Rakenne	29
4.2 Tiedonsiirron osapuolien tunnistaminen	30
4.3 Viestin tunnistaminen	31
4.4 Virnehallinta	31
4.4.1 Virheen tunnistaminen	31
4.4.2 Tilanhallinta	32

	2
4.5 Tiedonsiirto.....	33
5 Kehitysympäristöt ja -välineet	36
5.1 Kehitysalustan valinta.....	36
5.2 Espressif ESP32	37
5.3 Semtech SX1276	38
5.4 Ohjelmointikielet ja -työkalut.....	41
6 Verkkoprotokollan toteutus.....	42
6.1 Viesti	42
6.1.1 Viestin ylätunniste	43
6.1.2 Viestin runko.....	44
6.2 Paketti.....	44
6.2.1 Paketin ylätunniste	44
6.2.2 Paketin runko.....	45
6.3 Tietoliikenne	45
6.3.1 Toimintaperiaate	45
6.3.2 MessageStore	46
6.3.3 UplinkBuffer.....	46
6.3.4 DownlinkBuffer	47
6.3.5 SanlaProcessor.....	50
7 Testitapaukset ja tulokset.....	51
7.1 Skenaario A: Viestin suora lähetys laitteelta toiselle.	54
7.2 Skenaario B: Viestin välillinen lähetys laitteelta toiselle.....	56
7.3 Poikkeusskenaario: Paketti on jäänyt saapumatta perille.	59
8 Pohdinta.....	61
8.1 Prototyyppilaitteet	61
8.2 Vaatimusten toteutuminen.....	61
8.2.1 Toiminnallisuus.....	61
8.2.2 Tietoturva	61
8.2.3 Suorituskyky.....	62
8.2.4 Skaalautuvuus.....	62
8.3 Protokollan ongelmat ja jatkokehitys	63

	3
8.4 Yhteenveto	64
Lähteet	66
Liitteet	70

Kuviot

Kuvio 1. Viestin suora lähetys laitteelta toiselle.....	9
Kuvio 2. Viestin välillinen lähetys laitteelta toiselle	10
Kuvio 3. OSI-malli	12
Kuvio 4. LoRa-spektri	16
Kuvio 5. Kaistanleveys.....	19
Kuvio 6. Tähtitopologinen verkko.....	22
Kuvio 7. Täysi mesh-topologinen verkko.....	24
Kuvio 8. Osittainen mesh-topologinen verkko	24
Kuvio 9. TCP-paketti.....	26
Kuvio 10. UDP-Datagrammi	28
Kuvio 11. Opinnäytetyön protokolla.....	30
Kuvio 12. Tiedonsiirto kahden laitteen välillä	34
Kuvio 13. Korvaavan paketin pyytäminen viallisen tilalle	35
Kuvio 14. Puuttuvan paketin pyytäminen	36
Kuvio 15. SX1276 tärkeimmät ominaisuudet	39
Kuvio 16. Viestin rakenne	43
Kuvio 17. Viestin header-elementti.....	43
Kuvio 18. Paketin rakenne	44
Kuvio 19. Paketin ylätunniste	45
Kuvio 20. Saapuvien pakettien prosessointi.....	46
Kuvio 21. DownlinkPacketin rakenne	48
Kuvio 22. DownlinkPacketin hyötykuormapuskuri.....	48
Kuvio 23. DownlinkBufferin DownlinkPacket-puskuri.....	49
Kuvio 24. DownlinkBufferin toimintaperiaate.....	49
Kuvio 25. Saapuvan paketin vastaanotto	50

Kuvio 26. Saapuvan hyötykuorman vastaanotto.....	51
Kuvio 27. Testikäyttöön suunniteltu käyttöliittymä	52
Kuvio 28. Yhteystestin lähetetty paketti	52
Kuvio 29. Yhteystestin paketin vastaanotto ja lähetys.....	53
Kuvio 30. Yhteystestissä vastaanotetun paketin pudotus verkosta	53
Kuvio 31. Skenaarion A laiteasetelma	54
Kuvio 32. Skenaariossa A lähetetyt paketit	55
Kuvio 33. Skenaariossa A lähetettyjen pakettien vastaanotto ja esitys.....	55
Kuvio 34. Skenaariossa A vastaanotettujen pakettien välitys.....	56
Kuvio 35. Skenaariossa A välitettyjen pakettien pudotus verkosta	56
Kuvio 36. Skenaarion B laiteasetelma	57
Kuvio 37. Skenaariossa B lähetetyt paketit	58
Kuvio 38. Skenaariossa B välitettävien pakettien vastaanotto ja lähetys	58
Kuvio 39. Skenaariossa B välitettyjen pakettien vastaanotto ja esitys	58
Kuvio 40. Poikkeusskenaariossa lähetetyt paketit	59
Kuvio 41. Poikkeusskenaariossa puuttuvan paketin pyytäminen	60
Kuvio 42. Poikkeusskenaariossa pakettipyynnön vastaanotto ja vastaus.....	60
Kuvio 43. Poikkeusskenaariossa puuttuneen paketin vastaanotto ja esitys.....	60

Taulukot

Taulukko 1. Hajontakertoimen vaikutus chipien määrään.....	17
Taulukko 2. Hajontakertoimen vaikutus symbolinopeuteen	20
Taulukko 3. Hajontakertoimen vaikutus bittinopeuteen	20
Taulukko 4. LoRa pakettien kuormakoot eri kaistanleveyksillä ja hajontakertoimilla	21
Taulukko 5. Yhteystestin viesti	52
Taulukko 6. Yhteystestin vastausviesti	53
Taulukko 7. Skenaarion A testiviesti	54
Taulukko 8. Skenaarion B testiviesti	57
Taulukko 9. Poikkeusskenaarion testiviesti	59

Käsitteet

FIFO (First-in, First-out)

Kyseessä on metodi, jolla kuvataan tietorakenteiden käyttöä. Metodin kuvaamalla tavalla esimerkiksi viestijonon puskurin ensimmäinen eli vanhin viesti prosessoidaan aina ensimmäisenä, kun viimeisin eli uusin viesti prosessoidaan viimeisenä (FIFO (First-In-First-Out) approach in Programming n.d.).

Flash-muisti

Liikkumattomia osia sisältävä, haihtumattoman muistin omaava piiri, joka ei tarvitse jatkuvaa virtaa säilyttääkseen tietoa. Pienen kokonsa ansiosta Flash -muistia käytetään usein laitteissa, jotka ovat kokonsa ja virrankulutuksen puolesta rajoittuneita, kuten erilaiset mikrokontrollerit, matkapuhelimet jne. (Tyson n.d.).

Hyötykuorma

Tietoliikenneprotokollissa jokainen verkkoon lähetetty datayksikkö sisältää ylätunnisteen ja runko-osan. Ylätunniste sisältää tarvittavat tiedot datan perille saattamiseksi, kun rungossa olevaa varsinaista viestin sisältöä kutsutaan hyötykuormaksi (engl. payload).

Lähetin-Vastaanotin

Kaksisuuntainen radiolaitte, johon on sisäänrakennettu radiolähetin ja -vastaanotin.

Mikroprosessori

Mikrokokoisen tietokoneen keskusyksikkö. Ohjelmoitava laite, joka toteuttaa aritmeettisiä ja loogisia operaatioita annetusta syötteestä ja kykenee keskustelemaan muiden laitteiden kanssa (Tewari n.d.).

Puoliaaltodipoliantenni

Antenni, joka on leikattu ja käännetty keskeltä osoittamaan eri suuntiin. Antennien puolikkaat ovat pituudeltaan yhteensä puolet käytetystä aallonpituudesta. Puoliaaltodipoliantenni on suosituin radiolaitteissa käytettävä antennityyppi (Half Wave Dipole Antenna / Aerial n.d.).

Solmulaite

Tietoliikenteessä solmulaite on fyysinen laite, joka on kiinnittynyt käytettävään verkkoon. Solmulaite toimii verkon rajapintana tai muuna verkon liikennettä ohjaavana laitteena, kuten reitittimenä tai modeemina.

Tilakone

Ohjelmistokehityksessä termillä viitataan konseptiin, jossa laite tai olio kykenee esimääritellyn logiikan mukaisesti muuttamaan tilaansa. Tilakonetta käytetään ohjelmistojen logiikan määrittelyssä.

UUID

Kirjainlyhenne sanoista Universally Unique Identifier. UUID on 128-bittinen numero, jolla voidaan tunnistaa uniikki laite verkossa. Internetin standardisointiorganisaatio Internet Engineering Task Force (IETF) määrittelee UUID:n takaavan ainutlaatuisen tunnisteen: *"...and can guarantee uniqueness across space and time (RFC4122:2005, 1)"*. UUID versiossa 4 jokainen tavu generoidaan sattumanvaraisesti ilman suurempaa logiikkaa eroten esimerkiksi versiosta 1, joka luodaan mm. laitteen MAC - osoitteen ja prosessorin kellon perusteella.

RF (Radio Frequency)

Suomeksi puhutaan radioaaltojen taajuuksista. Kyseessä on elektromagneettisen säteilyspektrin oskillaation mitta. Radioaaltoja käytetään langattomaan datansiirtoon ja kommunikaatioon (Rouse n.d.).

Serial Peripheral Interface (SPI)

Usein mikrokontrollereiden käyttämä sarjaliikenneväylä mikrokontrollerin ja oheislaitteiden välistä kommunikointia varten.

Yhteydetön protokolla (engl. connectionless protocol)

Yhteydettömässä protokollassa tiedonsiirto aloitetaan ilman yhteyden muodostamista. Paketti lähetetään verkon yli kohdelaitteelle ilman varmuutta siitä, että viesti saadaan vastaanotettua (Gregg 2006).

1 Johdanto

Maaailmanlaajuinen verkon käyttöaste on ollut pitkään valtavassa kasvussa. Internet of Things -laitteiden vallatessa markkinoita ja sitä myöten verkkoon liittyneiden päätelaitteiden määrä on kasvanut räjähtävästi, mikä vaikuttaa tietoliikenteeseen monin tavoin. Stallings:n (2013, 21) mukaan jo vuoden 2016 aikana IP-verkkojen tietoliikenne kasvoi 372 eksatavusta 1.3 zettatavuun, noin 250 prosenttia. Tämän kaltainen verkkoliikenne kasvattaa kommunikaatioprotokollien suorituskykyvaatimuksia, jotka ovat edelleen ratkaisemattomia haasteita. Verkkoon kytkeytyneiden laitteiden määrän arvioidaankin ylittävän 25 biljoonaa vuoteen 2020 mennessä (Introduction to LoRa Technology – The Game Changer 2016).

Viime aikoina on alettu kiinnittää entistä enemmän huomiota vähävirtaisiin Internet of Things -laitteisiin ja sitä myöten vaihtoehtoisiiin, vähemmän virtaa kuluttaviin langattomiin tiedonsiirtotekniikoihin, jotka mahdollistavat entistä vapaamman laitesuunnittelun ja tasaavat internet-protokollan tietoliikennettä siirtämällä sitä muualle. Internet of Things -laitteiden yhdistävänä tekniikkana LoRa-verkon käyttö onkin kasvavassa suosiossa. LoRa on Semtech Corporationin kehittämä ja patentoima, radioaalloilla toimivaan ja vähävirtaiseen pitkän kantaman tiedonsiirtoon suunniteltu tekniikka (mt.).

Opinnäytetyön tarkoituksena oli tarkastella ja toteuttaa soveltuvuustutkimus (engl. feasibility study) langattoman, laaja-alaisen ja mesh-topologisen tiedonsiirtoverkon toteutuksesta LoRa-tekniikalla ja siihen soveltuvasta verkkoprotokollasta.

Toteutettavan verkon vaatimuksena oli, että dataa voitaisiin lähettää luotettavasti, vaikka osa verkon vastaanottavista solmulaitteista olisi lähettävän laitteen kantaman ulkopuolella. Useamman laiteryhmän tulisi kyetä kommunikoimaan rinnakkain omissa ryhmissään muiden, ryhmään kuulumattomien solmulaitteiden välityksellä. Tutkimustyö on luonteeltaan kehittämistutkimus.

Opinnäytetyön soveltuvuustutkimuksen prototyypilaitteet rakennettiin Xtensa-arkkitehtuuriin perustuvista Espressif ESP32 -mikrokontrollereista ja Semtech SX1276 LoRa-yhteensopivista RF lähetin-vastaanottimista.

2 Työn tavoitteet ja tutkimusongelma

2.1 Tutkimustavoitteet

Opinnäytetyössä tutkittiin LoRa-tekniikan soveltuvuutta eri laitteiden väliseen, monivaiheiseen langattomaan tiedonsiirtoon. Datan tulisi kulkea katkeamattomasti, vaikka tietoa vastaanottava laite ei olisi välittömässä yhteydessä lähettävään laitteeseen, vaan data kulki sitä välittävän solmulaitteen kautta.

Tutkimuksessa käytetylle protokollalle oli myös toissijaisen tärkeää mahdollistaa solmulaitteiden toiminta omina laitoryhminään. Laiteryhmien tulisi kyetä keskustelemaan muiden ryhmään kuuluvien solmulaitteiden kesken sisäisesti siten, etteivät ryhmän ulkopuoliset solmulaitteet esitä saapunutta tietoa loppukäyttäjälle, mutta kuitenkin välittäen saapunutta dataa muille verkkoon yhdistyneille laitteille.

Lopuksi tutkimuksen pohjalta toteutettiin prototyyppilaitteita, joilla tutkimuksen tulokset voitiin todentaa sekä todeta toteutuksen vahvuudet ja heikkoudet. Hyökyn ja Kyllösen (2013, 11) mukaan kehittämistutkimus on tutkimuksen laji, jossa pyritään yhdistämään teoreettiset ja kokeelliset vaiheet. Opinnäytetyötä voidaankin tutkivien ja toteuttavien piirteidensä vuoksi kutsua kehittämistutkimukseksi.

2.2 Tutkimuskysymykset

Opinnäytetyön tutkimuskysymyksiksi nousivat

1. Kuinka toteuttaa monivaiheinen, mesh-topologinen tiedonsiirto eri laitteiden välillä LoRa-verkossa?
2. Voiko olemassa olevia tiedonsiirtoprotokollia soveltaa LoRa-laitteiden välisessä tiedonsiirrossa?
3. Miten mahdollistaa useamman itsenäisen laitoryhmän kommunikaatio keskenään samalla taajuusalueella?

2.3 Toteutetut käyttöskenaariot

2.3.1 Skenaario A: Viestin suora lähetys laitteelta toiselle

Ensimmäisen skenaarion tarkoitus oli varmistaa tiedonsiirron toimivuus siten, että skenaarion testissä luotu viesti pystyttiin lähettämään monivaiheisesti, eli useammassa kuin yhdessä osassa. Skenaarion laiteasetelmassa kaksi solmulaitetta olivat suorassa yhteydessä toisiinsa (ks. kuvio 1).



Kuvio 1. Viestin suora lähetys laitteelta toiselle

Aktorit:

- Laite A: Viestin lähettäjä
- Laite B: Viestin vastaanottaja.

Käyttötapaus:

Laite A lähettää viestin laitteelle B.

Ennakkoehdot:

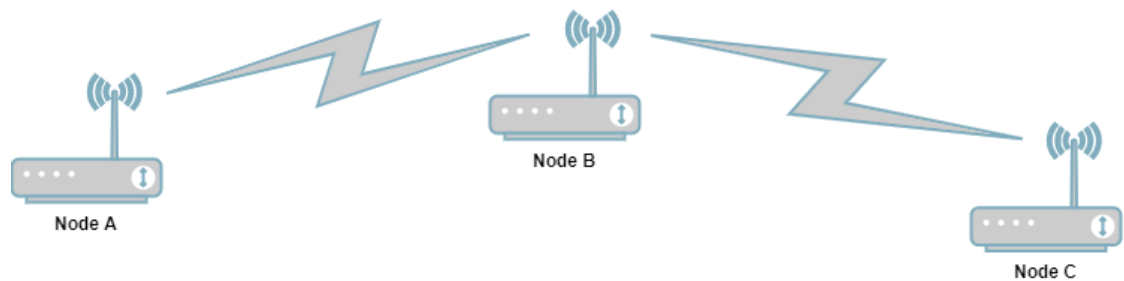
- Laite A ja laite B ovat molemmat rekisteröityneenä samaan laiteryhmään
- Lähetettävä viesti on pituudeltaan sellainen, ettei se mahdu lähetettäväksi yhdessä osassa, vaan tiedonsiirto on jaettava useampaan pakettiin.

Lopputila:

Viesti on vastaanotettu ja luettavissa laitteelta B.

2.3.2 Skenaario B: Viestin välillinen lähetys laitteelta toiselle

Toisen skenaarion tarkoitus oli varmistaa tiedonsiirron toimivuus siten, että tietoa pystyttiin lähettämään välillisesti ulkopuolisen solmulaitteen kautta ja monivaiheisesti kohdelaitteelle. Skenaarion laiteasetelmassa kaksi solmulaitetta kykenee keskustelemaan toisilleen vain kolmannen laitteen välityksellä (ks. kuvio 2)



Kuvio 2. Viestin välillinen lähetys laitteelta toiselle

Aktorit:

- Laite A: Viestin lähettäjä
- Laite B: Viestin välittäjä
- Laite C: Viestin vastaanottaja.

Käyttötapaus:

Laite A lähettää viestin laitteelle C laitteen B kautta.

Ennakkoehdot:

- Laite A ja laite C ovat molemmat rekisteröityneenä samaan laiteryhmään
- Laite B ei ole rekisteröityneenä ryhmään
- Laite A ja C eivät ole keskenään lähetyskantaman sisäpuolella
- Laite B on laitteen A lähetyskantamalla
- Laite C on laitteen B lähetyskantamalla
- Lähetettävä viesti on pituudeltaan sellainen, ettei se mahdu lähetettäväksi yhdessä osassa, vaan tiedonsiirto on jaettava useampaan pakettiin.

Lopputila:

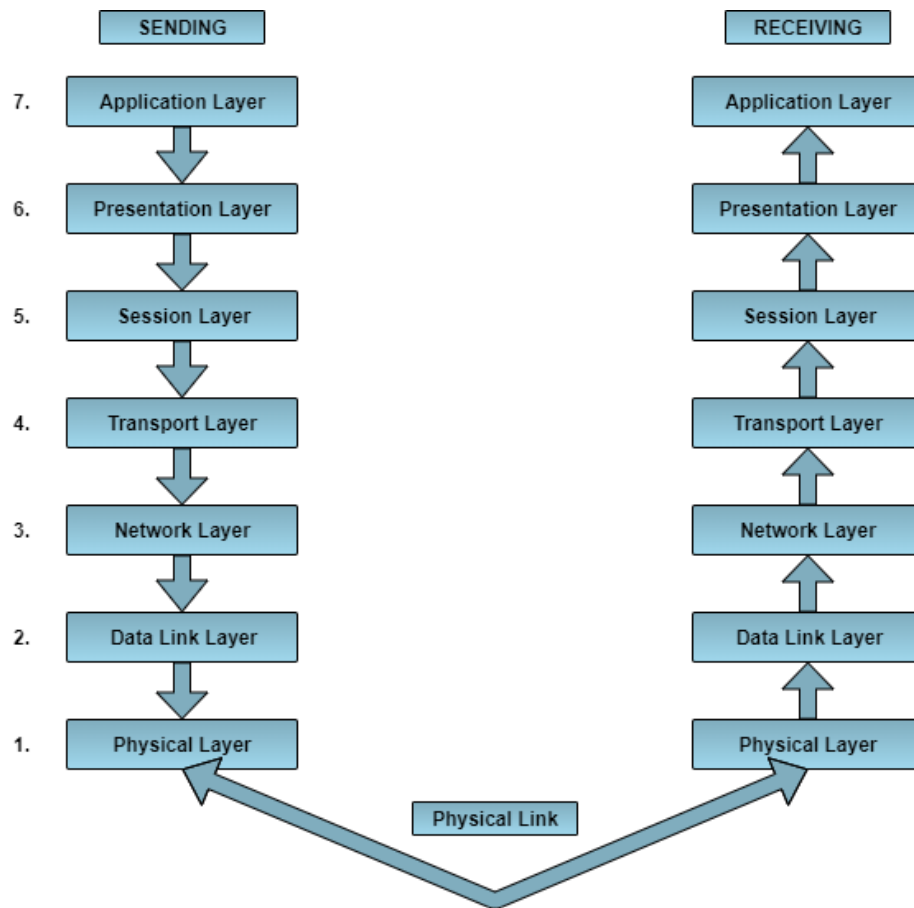
Viesti on vastaanotettu ja luettavissa laitteelta C. Laite B ei pysty lukemaan viestiä.

3 Langaton tiedonsiirto ja verkkoprotokollat

3.1 Yleistä

Verkkoprotokollaksi kutsutaan kokoelmaa sääntöjä ja tapoja, jotka mahdollistavat laitteiden välisen kommunikaation. Verkkoprotokolla määrittelee kuinka laitteet tunnistavat ja muodostavat yhteyden toisiinsa, sekä missä formaatissa data pakataan lähteviin ja saapuviin viesteihin (Mitchell n.d.).

Open Systems Interconnection Reference Model, lyhennettynä OSI-malli, on ISO:n (International Organization for Standardization) määrittelemä seitsemän kerroksen abstrakti tiedonsiirtoprotokollamalli, joka määrittelee ja standardisoi tiedonsiirron laitteiden välillä (ks. kuvio 3).



Kuvio 3. OSI-malli (Reference Models in Communication Networks n.d., muokattu)

Bahl:n (2018) mukaan OSI-mallilla voidaan havainnollistaa ja seurata datan liikkumista järjestelmissä, ottamatta kuitenkaan huomioon sen taustalla olevaa sisäistä arkkitehtuuria ja tekniikkaa, jolloin tiedonsiirtokokonaisuuden varsinaisen toteutuksen määrittely ja kehitys jää verkkoarkkitehtien vastuulle.

Kun solmulaite ottaa verkossa kulkevan datan vastaan, siirtyy data silloin fyysiseltä kerrokselta (optinen tiedonsiirto, radiosignaali, jne.) siirtokerrokselle, jossa saapunut data käännetään koneluettavaan muotoon biteiksi ja lähetetään eteenpäin verkkokerrokselle.

Verkkoprotokollat asettuvat OSI-mallissa tasolle 3: verkkokerros. Kerroksessa on neljä funktiota, joiden päätehtävä on ohjata verkossa kulkeva tai lähetettävä data päämääräänsä, jotka ovat

- Loogisen verkko-osoitteen kääntäminen fyysiseksi osoitteeksi
- Verkossa kulkevan datan reititys
- Vuonhallinta, vianhallinta ja segmentointi
- Datat pilkkominen pienempiin paketteihin.

Yksi verkkoprotokollan yleisimmistä ongelmista on määrittely, kuinka paketit saadaan välitettyä lähteestä kohteeseensa, sillä reitit voivat olla staattisia ja lähes muuttumattomia, tai täysin dynaamisia ja jatkuvasti muuttuvia. Muita ongelmakohtia ovat verkon ylikuormituksen aiheuttamat pullonkaulat yksittäisissä solmulaitteissa ja verkon palvelun laatu, eli viive, lähetysaika jne.

Verkkokerrokselta saapuva paketti siirretään verkkokerrokselta kuljetuskerroksen tietoliikenneprotokollille, joiden tehtävät ovat ISO-mallissa lähes samat kuin ylemmällä verkkokerroksella. Kuljetuskerros huolehtii pakettien rakentamisesta valmiiksi viesteiksi, vuonhallinnasta ja virnehallinnasta. Myös kuljetuskerros kärsii samoista ongelmista kuin verkkokerros (Reference Models in Communication Networks n.d.). Usein verkko- ja kuljetuskerrosten toteutukset eivät ole selkeästi eroteltavissa toisistaan, vaan implementoivat molemmat kerrokset yhtenä kokonaisuutena, pinona.

3.2 Langaton tiedonsiirto

Langaton tiedonsiirto ei periaatteeltaan eroa muilla tavoin toteutettavasta tiedonsiirrosta, vaan tekniikoiden erottava tekijä on OSI-mallissa kuvatussa fyysisessä kerroksessa tapahtuva digitaalisen signaalin muuttaminen ja lähettäminen fyysisesti liikutettavaan muotoon sekä sen vastaanotto ja muuttaminen takaisin digitaaliseksi, koneluettavaksi signaaliksi. Yksi langattoman tiedonsiirron tavoista on sen toteuttaminen radioaaltojen avulla. Tällä tavoin fyysisesti liikutettavan signaalin lähettäjä tarvitsee aina laitteen, jolla radiosignaali tuotetaan, eli radiolähtetimen. Radiolähtetin lähettää radiosignaaleja kohti siihen suunniteltua laitetta, radiovastaanotinta.

Radioverkossa tapahtuvalla tietoliikenteellä on kuitenkin omat haittapuolensa ja rajoitteensa, sillä radiolaitteet ovat alttiita ulkoisille häiriötekijöille, kuten muiden radiolaitteiden lähettämille radiosignaaleille.

On tärkeää, että radiolaitteet käyttävät sille tarkoitettua taajuutta. Jotta radiolaitteet eivät häiritsisi muita radiolaitteita tai häiriintyisi muista radiolaitteista, niiden tulee täyttää niille EU-alueella asetetut vaatimukset (Radiolaitteiden vaatimustenmukaisuus takaa toimivan radioliikenteen 2019).

Radioaallot voivat aiheuttaa myös terveyshaittoja, sillä Säteilyturvakeskuksen (Radioaallot ympäristössämme 2009) mukaan osa radioaaltojen kuljettamasta energiasta jää antennin lähellä oleviin ihmisiin aiheuttaen lämmön nousua kudoksissa. Nyberg ja Jokela (2006, 320) huomauttavatkin, että riskin kantajan painotetaan hankkivan riskin hallintaa varten tarvittavat tiedot ja taidot, riskin kantajan ollessa täydessä vastuussa turvallisuudesta. Siksi oli tärkeää selvittää, ettei työssä käytettyjen radiolaitteiden säteilyteho ylittänyt määrättyjä arvoja.

Säteilytehosta puhutaan käsitellessä lähettimen ulostulosta mitattavaa signaalin tehoa. Useimmiten markkinoilla liikkuvat laitteet dokumentoidaan juuri tällä säteilyteholla, mutta se jättää kertomatta osan totuudesta, sillä varsinkin antennista mitattava säteilyteho eroaa usein dokumentoidusta. Lähettimen antennista mitattavaa säteilytehoa kutsutaankin efektiiviseksi säteilytehoksi (ERP, engl. Effective Radiated Power). Termi on IEEE:n standardisoima määritelmä lähettävän radiolaitteen kokonaissäteilytehosta puolialtodialiantennista mitattuna, jossa säteilytehosta on otettu huomioon lähettimen antennikaapelista ja liittimistä johtuvat tehonmenetykset sekä antennin tuottama vahvistus. ERP-arvolla voidaan laskea eri antennityyppien vaikutusta laitteesta mitattavaan säteilytehoon helpottaen radioverkkojen suunnittelua (Silver 2018).

Liikenne- ja viestintäviraston määräyksestä LoRa-laitteen suurin sallittu efektiivinen säteilyteho on 25 mW ERP ja suurin sallittu toimintasuhde 1% (15 AO/2019 M, 10 §). Toimintasuhde tarkoittaa lähettimen suhteellista lähetysaikaa yhden tunnin jaksossa,

jolloin esimerkiksi jokaisen tunnin aikana radiolaite saa lähettää yhteensä 36 sekuntia.

3.3 LoRa

3.3.1 Perusteet

LoRa, lyhenne sanoista Long Range, on Semtech:n kehittämä chirp-hajaspektri modulaatiotekniikan johdannainen, joka on suunniteltu langattomaan ja vähävirtaiseen pitkien matkojen tiedonsiirtoon radioverkossa, ja josta on tullut yksi varteenotettavimmista Internet of Things-verkkoteknologioista maailmanlaajuisesti (What is LoRa®? n.d.).

LoRa-tekniikalla toteutettava tiedonsiirto pohjautuu radioaaltojen taajuuden modulaatioon ja sitä kautta fyysisesti liikutettavaan datasiignaaliin. Tällaista tekniikkaa kutsutaankin taajuusmodulaatioksi, jossa lähetettävän analogisen tai digitaalisen signaalin aaltoa muutetaan lähetettävän datan perusteella. Kun lähetävä ja vastaanottava laite toimivat samalla taajuusmodulaatiolla, lähetettävä data voidaan kääntää radiosignaaliksi ja siitä takaisin tietokoneen luettavaksi dataksi.

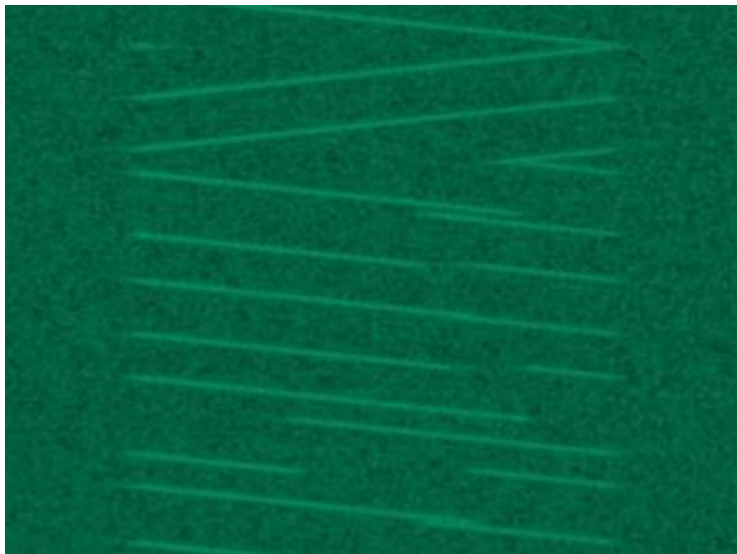
Tietoliikenne- ja verkkoprotokolla suunniteltiin tukemaan tiedon lähettämistä pitkien matkojen päähän solmulaitteisiin, jotka eivät välttämättä ole suorassa yhteydessä lähetävään laitteeseen, vaan lähetettävän tiedon oli kuljettava kolmannen tai useamman solmulaitteen välityksellä. Verkon laitteet suunniteltiin toteutettavaksi myös niin, että verkkoa voidaan kasvattaa liittämällä siihen laitteita omina ryhminään siten, että viestit näkyvät vain lähetävän laitteen ryhmän jakaville solmulaitteille kaikkien osallistuessa kuitenkin viestin jakamiseen verkossa.

LoRa-arkkitehtuurissa määritellään suurimmat sallitut pakettien koot verrattain pieniksi. Tärkeimmät datansiirtoon ja pakettikokoon vaikuttavat arvot ovat hajontakerroin (engl. spreading factor), kaistanleveys (engl. bandwidth) ja Coding Rate. Nämä arvot muodostavat yhdessä bittinopeuden (engl. bitrate), jolla tietoa saadaan kuljetettua verkon ylitse.

3.3.2 Chirp

Chirp on kirjainlyhenne sanoista Compressed High Intensity Radar Pulse. Kyseessä on protokolla, joka on alun perin kehitetty toisen maailmansodan aikana luotaus- ja tutkakäyttöön, joten kyseessä ei ole mitenkään uusi tekniikka (Data Rate and Spreading Factor n.d).

Chirp tarkoittaa signaalin hetkeä, ”pulsia”, jolla on lineaarinen muuttumaton amplitudi vaihtelevalla taajuudella. Taajuuden noustessa signaalia kutsutaan up-chirpiksi ja taajuuden laskiessa signaalia kutsutaan down-chirpiksi (Hamrioui, Monacruz, Mroue, Nasser, Parrein & Rouyer 2018, luku III). Kuviossa 4 esitetään LoRa-lähetyksestä mitattu spektri eli signaalin taajuusjakauma, josta on havaittavissa molemmin suuntaiset chirpit.



Kuvio 4. LoRa-spektri (Knight n.d.)

LoRa-tekniikka lähettää käyttämänsä chirpit alun perin 1940-luvulla tutkakäyttöön kehitetyllä teknologialla (Ianelli 2003) nimeltään Chirp Spread Spectrum. Kyseessä on modulaatiostandardi, jolla kyetään lähettämään dataa radioaalloilla pitkien matkojen päähän 868MHz taajuudella (Pollin & Reynders 2016). Standardi tarjoaa vastustusta taajuusselektiiviselle kohinalle ja häirinnälle laskeneen spektritehokkuuden kustannuksella (mt. luku III).

3.3.3 Spreading Factor

Chirp Spread Spectrum -modulaatiotekniikassa ajallisesti pienintä mitattavaa hetkeä, jossa moduloidun signaalin parametrit pysyvät muuttumattomina, kutsutaan chipiksi. Hajontakertoimella, lyhenteeltään SF, kuvataan kuinka monta chipia vaaditaan muodostamaan yksi symboli. Symboli kuvastaa siirrettyä datamäärää yhden Chirp Spread Spectrum -modulaation mukaisen chirpin aikana.

Mitä suurempi hajontakerroin, sitä enemmän chipeja yksi symboli vaatii. Symbolin sisältämä chipien määrä N voidaan laskea kaavalla 1.

$$N = 2^{SF} \quad (1)$$

LoRa-laitteet toimivat useimmiten hajontakertoimilla SF7 – SF12. Joitain poikkeuksia tähän kuitenkin on, kuten opinnäytetyössä käytetyn lähetin-vastaanottimen tuki hajontakertoimelle SF6 (SX1276/77/78/79 - 137 MHz to 1020 MHz Low Power Long Range Transceiver 2019, luku 4.1.1.2). Kaavasta voidaan laskea chipien määrät symboleissa eri hajontakertoimilla (ks. taulukko 1).

Taulukko 1. Hajontakertoimen vaikutus chipien määrään

Spreading Factor	Chips / Symbol
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096

Hajontakertoimen numeerinen arvo kertoo, kuinka monta bittiä symboli sisältää. Esimerkiksi SF6 kantaa 6 bittiä informaatiota, SF 7 kantaa 7 bittiä ja niin edelleen. Aiemmasta kaavasta (ks. kaava 1) voidaan nähdä jokaisen symboliin lisätyn bitin tuplaavan symbolin vaatiman chipien määrän. LoRa-tekniikassa tiedonsiirron nopeutta kutsutaan symbolinopeudeksi (engl. symbol rate), lyhenteeltään R_s . Vaikka

symboli voi sisältää muuttuvan määrän bittejä, eikä termi sinällään kuvasta varsinaista bittinopeutta, tästä voitiin kuitenkin tulkita datan siirron verkkoon olevan pienimmällä hajontakertoimella SF6 olevan ajallisesti nopein ja suurimmalla hajontakertoimella SF12 olevan ajallisesti hitain. Hajontakertoimen perusteella ei kuitenkaan vielä voida laskea tarkkaa tietyn datamäärän lähetykseen vaadittua aikaa, vaan siihen vaikuttavat myös signaalin kaistanleveys ja coding rate.

3.3.4 Coding Rate (CR)

LoRa-tekniikassa lähetettävän datan eteen lisätään virheitä korjaavaa signaalia FEC (Forward Error Correction), jotta lähetettävä data säilyisi mahdollisimman eheänä. FEC:n idea on lisätä alkuperäisestä datasta tietyllä algoritmilla laskettua toistuvia bittejä, jotka auttavat vastaanottavaa laitetta palauttamaan häiriöstä korruptoitunutta lähetettyä dataa.

Coding ratella viitataan suhteeseen hyötykuorman bittimäärästä. Arvo on murtoluku $\frac{4}{4+n}$ jossa osoittaja on kiinteä 4 ja nimittäjä $4+n$, jonka arvo n on väliltä 1-4. Coding ratea kutsutaankin usein suoraan n arvolla. Esimerkiksi hajontakerroin SF11 sisältää 11 bittiä. Data lähetetään CR arvolla 3, jolloin lähetetystä bittimäärästä voidaan laskea varsinaisen hyötykuorman ja FEC-datan määrät (ks. kaava 2).

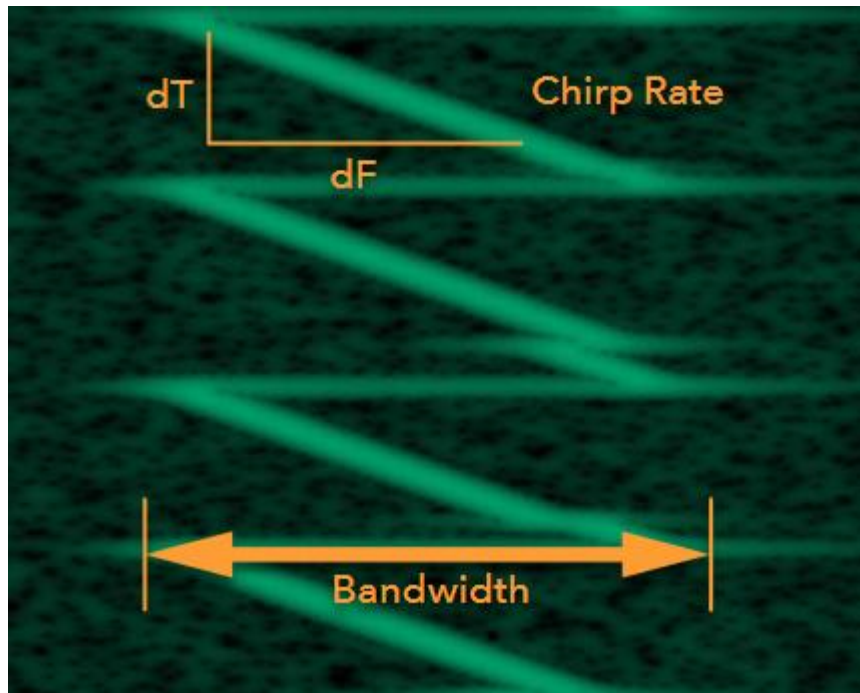
$$CR3 \left(\frac{4}{4+3} \right), 11 * \frac{4}{7} = 6,3 \text{ bits} \quad (2)$$

Laskettuna jokainen lähetetty 11 bittiä sisältää 6,3 bitin verran varsinaista hyötykuormaa ja 4,7 bittiä virheenkorjausdataa.

Coding raten kasvattaminen siis johtaa kasvaneeseen dataliikenteeseen ja pienentyneeseen efektiiviseen bittinopeuteen, mutta helpompaan korruptoituneen datan palautukseen. Opinnäytetyössä oli siis hyvä aloittaa oletusarvoisesti pienimmästä CR-arvosta ja kasvattaa sitä, jos se nähtiin tarpeelliseksi.

3.3.5 Kaistanleveys

Radiotekniikassa kaistanleveys (engl. bandwidth), lyhenteeltään BW, tarkoittaa radioaallon laskevan ja nousevan rajataajuuden erotusta, taajuusaluetta (ks. kuvio 5). Kaistanleveyttä kuvataan useimmiten hertseinä (Hz).



Kuvio 5. Kaistanleveys (Knight n.d.)

LoRa-tekniikassa tämä taajuusalue tarkoittaa yhtä chipeja lähettävää chirpia. Kaistanleveys siis määrittelee, kuinka monta chipia kyetään lähettämään sekunnissa. Esimerkiksi 125 kHz:n kaistanleveydellä laite lähettää 125 000 chipia sekunnissa, jolloin yhden chipin lähetys vie 480 μ s.

Tietämällä kaistanleveys voidaan laskea hajontakertoimien muodostamat symbolinopeudet (ks. kaava 3).

$$R_s = \frac{BW}{2^{SF}} \quad (3)$$

Kaavan avulla esimerkkinä käytetyllä 125 kHz:n kaistanleveydellä laskettiin seuraavat hajontakertoimien symbolinopeudet (ks. taulukko 2)

Taulukko 2. Hajontakertoimen vaikutus symbolinopeuteen

Spreading factor	Symbol rate (symbols/s)
6	1953.13
7	976.563
8	488.281
9	244.141
10	122.07
11	61.0352
12	30.5176

Kun symbolinopeus tiedetään, voidaan seuraavalla kaavalla laskea lähetyksen bittinopeus (engl. bitrate) R_b , eli kuinka nopeasti dataa voidaan lopulta siirtää verkkoon (ks. kaava 4) (Grilo 2018).

$$R_b = SF * R_s * CR \quad (4)$$

Kaavan 4 CR-arvona käytettiin opinnäytetyön toteutusvaiheessa käytettävän Arduino LoRa -kirjaston määrittelemää oletusarvoa $\frac{4}{5}$ (Mistry n.d.). Kaavan avulla voitiin taulukoida varsinaiset bittinopeudet eri hajontakertoimilla (ks. taulukko 3).

Taulukko 3. Hajontakertoimen vaikutus bittinopeuteen

Spreading factor	Bitrate (bits/s)
6	9375.02
7	5468.75
8	3125
9	1757.82
10	976.56
11	537.11
12	292.969

Vaikka suurempi kaistanleveys ja hajontakerroin sallivat molemmat suuremman efektiivisen datanopeuden, niiden kasvattaminen heikentää vastaanottavan laitteen herkkyyttä (SX1276/77/78/79 - 137 MHz to 1020 MHz Low Power Long Range Transceiver 2019, luku 4.1.1.4). Herkkyyden menetys vuorostaan johtaa lyhempiin lähetysetäisyyksiin, jos vaatimuksena on, että lähetettävä datan tulee pysyä täysin eheänä.

Toiminnallisuuden kannalta optimoitavaksi voidaankin valita vain yksi metriikka pitkän kantaman, korkean suorituskyvyn ja nopean lähetyksen joukosta. Yhden metriikan optimointi uhraa muita (Introduction to LoRa n.d.). Taulukossa 4 kuvataan datan lähetyksenopeutta ja suurimpia sallittuja pakettikokoja eri kaistanleveyksillä ja hajontakertoimilla.

Taulukko 4. LoRa pakettien kuormakoot eri kaistanleveyksillä ja hajontakertoimilla (mt.)

Spreading Factor & Bandwidth	Transmit Data rate	Maximum Uplink Payload Size	
		North America	Europe
SF_8 500kHz	12.5 kbps	242 bytes	-
SF_7 125kHz	5.47 kbps	242 bytes	242 bytes
SF_8 125kHz	3.125 kbps	129 bytes	242 bytes
SF_9 125kHz	1.76 kbps	53 bytes	115 bytes
SF_10 125kHz	0.98 kbps	11 bytes	51 bytes
SF_11 125kHz	0.44 kbps	-	51 bytes
SF_12 125kHz	0.25 kbps	-	51 bytes

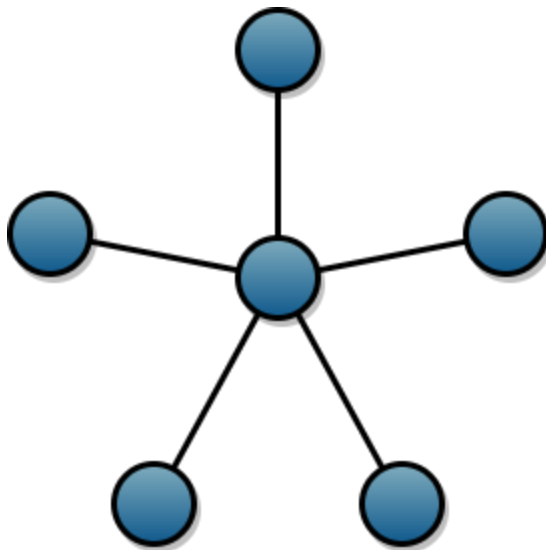
Mitä pidemmälle tietoa siirretään langattomasti, sitä enemmän datanopeudesta on siis tingittävä. Opinnäytetyössä keskityttiin pienikokoisen datan siirtämiseen luotettavasti pitkien matkojen päähän, jolloin datanopeus jäi toissijaiseksi ja lähetyksen vastaanotin voitiin konfiguroida lähettämään tietoa häiriövapaammin ja luotettavammin, kuitenkin jättäen vielä hieman säätövaraakaan skaalan päähän. Siksi työssä käytettävän signaalin hajontakertoimeksi valikoitui SF11 ja kaistanleveydeksi

125 kHz. Taulukon 4 suositusten mukaisesti työssä käytettävän datapaketin suurimmaksi sallituksi kooksi tuli 51 tavua.

3.4 Verkon rakenne

Verkkotopologian tarkoitus on kuvailla, kuinka tietokoneet tai muut solmulaitteet ovat yhdistyneinä verkossa. Langallisissa tiedonsiirtojärjestelmissä verkkotopologialla kuvailtaisiin fyysisiä datakaapeleita, solmulaitteita ja reittejä solmulaitteiden välillä. Langattomissa tiedonsiirtojärjestelmissä tarkoitetaan virtuaalista solmulaitteiden asetelmaa ja sitä, miten solmulaitteet kommunikoivat keskenään tässä asetelmassa.

Esimerkiksi jos työssä toteutettava laite siirtäisi dataa muille lähetyksentamalla oleville laitteille, mutta kantamalla olevat laitteet eivät keskustele toistensa kanssa suoraan tai ollenkaan, kyseessä olisi niin sanottu tähtitopologinen verkko. Näin ollen dataa siirtävä laite toimisi ainoana tukiasemana verkon solmukohtassa (ks. kuvio 6).



Kuvio 6. Tähtitopologinen verkko

Kyseisen verkon termein dataa lähettävä laite toimisi verkossa keskuslaitteena, jonka kautta kaikki liikenne kulkisi. Tähtitopologia on ääriesimerkki yksinkertaisesta verkosta, joka ei kuitenkaan soveltunut opinnäytetyön vaatimuksiin. Työn yhtenä tavoitteena oli toteuttaa mahdollisimman laaja-alainen tiedonsiirtoverkko, joka

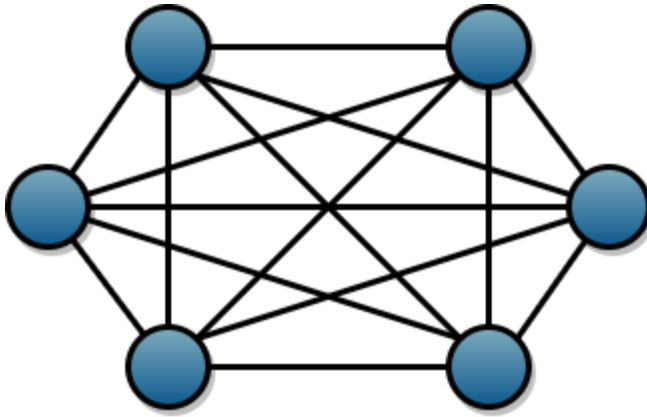
tähtitopologisesti toteutettuna jäisi pinta-alaltaan riippuvaiseksi lähetävän solmulaitteen suorituskykyyn ja sijoitteluun. Keskimääräinen LoRa -laite kykenee hyvin sijoiteltuna noin 2-3 kilometrin lähetyspinta-alaan kaupunkialueella ja noin 5-7 kilometriin maaseudulla (Chauhan & Lee 2018), jolloin verkon kokonaispinta-ala jäisi verrattain pieneksi eivätkä työn vaatimukset toteutuisi.

Tiedonsiirtoverkon kasvattamiseksi työssä toteutettava laite ja protokolla tuli suunnitella siten, että viestin vastaanottaneet laitteet voisivat myös toimia ns. uusina keskuslaitteina ja välittää viestiä kaikille uusille, alkuperäisen lähettäjän kantaman ulkopuolella oleville laitteille. Verkko olisi tällöin mesh-topologinen verkko.

Mesh-verkosta puhuttaessa käsitellään verkkotopologiaa, jossa verkon jokainen solmulaite, "node", on itsenäisesti yhteydessä verkon muihin solmulaitteisiin ilman yhteistä tiedonsiirtokeskusta (Shekhar 2016). Mesh-topologinen verkko voidaan jakaa kahteen lohkokoon:

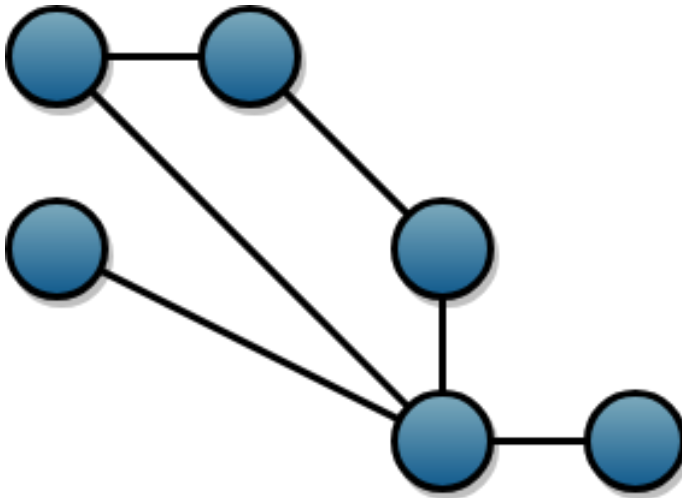
- Täysi mesh-topologia (fully connected)
- Osittainen mesh-topologia (partially connected).

Täydessä mesh-topologisessa verkossa jokainen verkon solmulaite toimisi tukiasemana ja ollen suorassa yhteydessä toisiinsa (ks. kuvio 7). Tämä verkkotopologia ei työssä kuitenkaan toteutunut, sillä tiedonsiirron langattomuuden vuoksi ei ole takeita, että kaikki verkon solmulaitteet olisivat toistensa lähetykskantamalla.



Kuvio 7. Täysi mesh-topologinen verkko

Osittaisesta mesh-topologiasta puhuttaessa verkon solmulaitteiden ei tarvitse olla suorassa yhteydessä kaikkiin muihin solmulaitteisiin, vaan siirrettävä data pystyy hyppimään laitteelta toiselle luotettavasti, vaikka osa solmulaitteista olisi lähettävän solmulaitteen kantaman ulkopuolella (ks. kuvio 8).



Kuvio 8. Osittainen mesh-topologinen verkko

Opinnäytetyössä toteutetun tiedonsiirtoverkon ollessa langaton solmulaitteiden jatkuvaa suoraa yhteyttä toisiinsa ei voida varmasti todentaa, joten verkkoa voidaan kuvailla tyypiltään osittaiseksi mesh-topologiseksi verkoksi.

3.5 Tunnetut protokollat

3.5.1 Protokollien valinta

Tiedonsiirron toteutusta varten tutkittiin ja vertailtiin olemassa olevien tietoliikenneprotokollien soveltuvuutta opinnäytetyön toteutuksessa.

Tietoliikenneprotokollia on olemassa lähes mittaamaton määrä, mutta tutkimuksen selkeyden nimissä syvempään tarkasteluun tietoliikenneprotokollista valittiin kaksi tunnetuinta, TCP ja UDP -tietoliikenneprotokollat.

3.5.2 TCP

TCP on kirjainlyhenne sanoista Transmission Control Protocol. Protokolla on luotu takaamaan luotettava tiedonsiirto verkon yli, joka varmistaa seuraavat asiat:

- Kaikki paketit saapuvat kohteeseensa, eikä yksikään katoa matkalla
- Paketit saapuvat oikeassa järjestyksessä
- Lähetyksen viive ei kasva liian suureksi. Tällä varmistetaan esimerkiksi musiikintoiston katkeamattomuus.

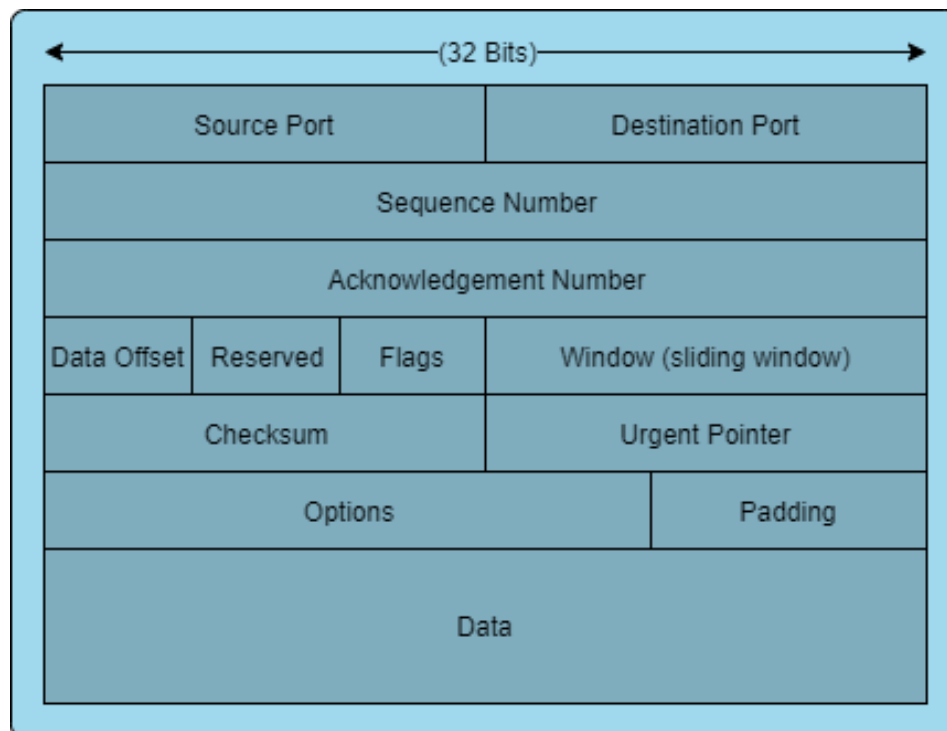
Unuth:n (2018) mukaan näillä varmennuksilla voidaan luottaa lähetetyn tiedon johdonmukaisuuteen, järjestykseen ja täydellisyyteen.

TCP-protokollan tiedonsiirtokonsepti pohjautuu yhteen, katkeamattomaan ja määrittelemättömän pituiseen datavirtaan (engl. stream), jossa lähetettävä data jaetaan sattumanvaraisesti segmentteihin (Hanson 2011). Datavirtaa ei kuitenkaan katkaista jokaisen lähetetyn viestin jälkeen, vaan viimeisimmän lähetetyn segmentin saavuttua perille.

Tiedonsiirto TCP-internetprotokollassa suoritetaan kolmessa osassa:

1. Yhteyden muodostaminen. Lähettävä ja vastaanottava laite muodostavat yhteyden kolmitiekättelyllä. Kolmitiekättelyssä:
 - a) Lähettävä laite lähettää SYN-paketin (Synchronization)
 - b) Vastaanottava laite vastaa SYN/ACK-paketilla (Acknowledge) merkiksi saapuneesta SYN-paketista
 - c) Lähettävä laite lähettää ACK-paketin merkiksi saapuneesta SYN/ACK-paketista
2. Tiedonsiirto. Lähtevät paketit sisältävät varsinaisen kuorman lisäksi TCP:n vaatimat lisätiedot, kuten paketin sekvenssinumeron ja muut tunnistimet. Vastaanottava laite vastaa jokaiseen lähetettyyn pakettiin ACK-paketilla.
3. Yhteyden sulkeminen:
 - a) Lähettävä laite lopettaa lähetyksen FIN-paketilla (Finish)
 - b) Vastaanottava laite vahvistaa lopetuksen FIN/ACK-paketilla
 - c) Lähettävä laite vastaa ACK-paketilla saapuneesta FIN/ACK-paketista.

TCP-paketin ylätunniste sisältää varsinaisen kuorman lisäksi 12 erikokoista kenttää yhteensä 160 bitin, eli 20 tavun verran, ilman options-tietojen laskemista mukaan (ks. kuvio 9).



Kuvio 9. TCP-paketti

TCP:n luotettavuus perustuu kolmivaiheiseen virheenkorjaukseen.

Virheenkorjauksen ensimmäinen vaihe on paketin tarkistussumman (engl. checksum) laskenta ja vertailu. Jokainen segmentti sisältää 16-bittisen tarkistussumman, jonka lähettävä osapuoli laskee segmentin hyötykuormasta. Vastaanottava osapuoli laskee saapuneesta segmentista vastaavan arvon ja jos tiedonsiirron molempien osapuolien laskemat tarkistussummat eivät täsmää, vastaanottava osapuoli pudottaa saapuneen segmentin. Tällöin segmentti tulkitaan kadonneeksi, eikä sitä lisätä muun saapuneen datan joukkoon (THIRU n.d.).

Virnehallinnan toinen vaihe on vastaanotetun segmentin kuittaus (engl. acknowledgement). Vastaanottajan tulee kuitata onnistuneesti saapuneet segmentit ACK-paketilla. Kuittaukseen ei lähetetä, jos segmentin tarkistussummat eivät täsmää (mt.).

Kolmas vaihe on segmentin uudelleenlähetyks (engl. retransmission). TCP-protokolla implementoi Retransmission Time-Out (RTO) -ajastimen kaikille lähetetyille segmenteille. Ajastimen toimintaperiaate on mitata aikaa segmentin lähetyksestä vastaanottajan kuittauksen saapumiseen. Jos lähetetty segmentti on korruptoitunut, kadonnut tai viivästynyt ja ajastimeen määritelty aika kuluu loppuun, lähetetään kyseinen segmentti uudelleen. RTO-ajastimen määritelty kesto muuttuu dynaamisesti keskimääräisen segmentin lähetyksen ja sen kuittaukseen kuluneen aikaeron mukaan (mt.).

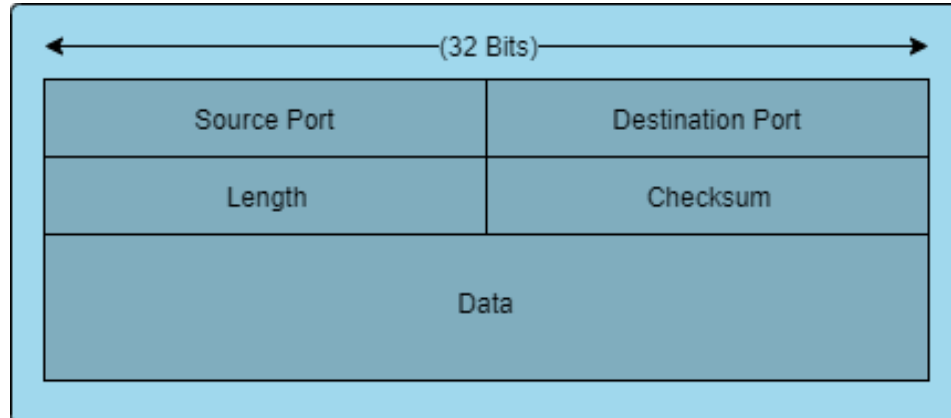
TCP-protokollaa on useimmin käytetty osana IP-protokollaa (Internet Protocol). Estääkseen pakettia jäämästä verkkoon pyörimään ikuisesti, protokolla implementoi tavun mittaisen Time-to-Live -kentän (TTL). Kenttä toimii siten, että pakettia lähetettäessä sille annetaan järjestelmäriippuvaisesti alkuarvo 1-255, jota vähennetään jokaisella paketin hypyllä verkossa. Kun kentän arvo tippuu nolnaan, paketti poistetaan verkkoliikenteestä (Iveson 2019).

3.5.3 UDP

UDP on kirjainlyhenne sanoista User Datagram Protocol, yksi vanhimmista verkkoprotokollista ja usein ensimmäinen vaihtoehto TCP-protokollalle. UDP-protokolla on kehitetty pääasiassa sovelluksille, jotka vaativat lähes reaaliaikaista tiedonsiirtoa, kuten tietokonepelaaminen ja videokonferenssit, johon TCP:n suorituskyky ei riitä (Mitchell 2019).

UDP on yhteydetön (engl. connectionless) protokolla, jossa lähettävän ja vastaanottavan laitteen ei tarvitse muodostaa yhteyttä toisiinsa ennen lähetyksen aloittamista, eikä sulkea sitä lähetyksen valmistuttua toisin kuin TCP-protokollassa (Gregg 2006).

Yksittäistä UDP-pakettia kutsutaan datagrammiksi. Datagrammi sisältää varsinaisen kuorman lisäksi neljä kenttää, joista jokainen on 16 bittiä eli 2 tavua pitkä (ks. kuvio 10). Kokonaisuudessaan UDP-datagrammin ylätunniste vie 8 tavua.



Kuvio 10. UDP-Datagrammi

UDP-protokolla implementoi samalla algoritmilla lasketun tarkistussumman kuin TCP-protokolla, mutta koska UDP ei implementoi lähetyksen valvontaa kuitausviesteillä samoin kuin TCP, tai millään muulla tavalla, saapunut ja virheelliseksi tulkittu segmentti pudotetaan pois ilman jatkotoimenpiteitä. Tämä aiheuttaa mahdollisia epämuodostumia saapuneessa datassa.

Yksittäisten segmenttien pudottaminen ilman uudelleenryityksiä ja pakettien mahdollinen saapuminen eri järjestyksessä kuin ne lähettäessä olivat, sallii UDP-protokollalle paremman suorituskyvyn ja pienemmän viiveen kuin TCP-protokollassa. Tämä kutistaa myös tiedonsiirron vaatimaa kaistanleveyttä (Mitchell 2019) datan luotettavuuden kustannuksella.

4 Opinnäytetyön verkkoprotokolla

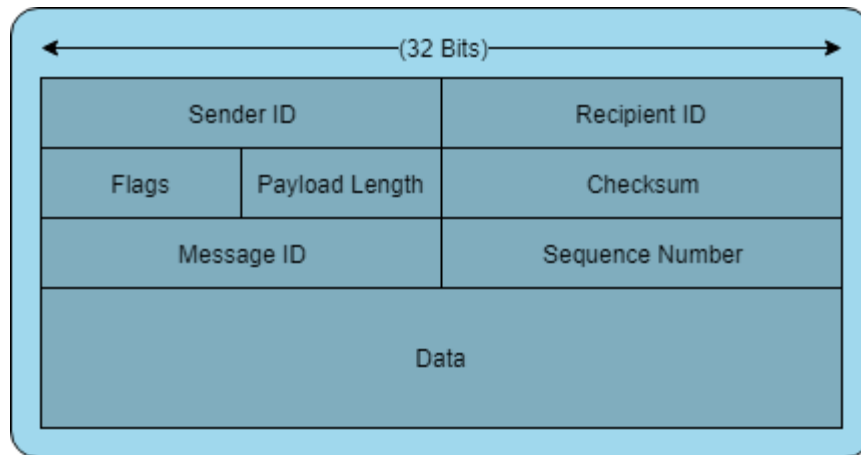
4.1 Rakenne

Vaikka TCP-protokolla virhesietoisuudellaan ja pakettien järjestyksestä huolehtivaisuudellaan oli houkutteleva vaihtoehto opinnäytetyössä käytettäväksi verkkoprotokollaksi, TCP-paketin ylätunniste sisälsi kuitenkin huomattavan määrän ylimääräistä tietoa opinnäytetyön tarpeisiin. TCP-paketin ylätunnistekoon ollessa 20 tavua varsinaiselle lähetettävälle kuormalle jäisi tilaa 31 tavun verran, eli yhdessä paketissa voisi kuljettaa vastaavan määrän merkkejä. Hyötykuorman pieni koko yhdistettynä TCP:n yhteydenmuodostukseen ja varmistusviesteihin kasvattaisi pienelläkin viestillä verkon kuormituksen ylettömän suureksi.

Verkon suuren kuormittavuutensa vuoksi kiinnostuttiin UDP-datagrammista protokollan yhteydettömyyden ja vain 8 tavun ylätunnistetietojensa koon vuoksi. 8 tavun ylätunnistetiedot sallisivat hyötykuorman kasvatuksen 43 tavuun, tai merkkiin, saakka.

UDP-datagrammia tutkiessa huomattiin, ettei tämäkään protokolla sopisi suoraan opinnäytetyöhön, ellei työn vaatimuksissa esimerkiksi listattua laitteiden jakamista laiteryhmiin siirrettäisi OSI-mallissa kuljetuserroksesta sovelluserrokseen. Edellä mainittu vaatisi hyötykuorman käyttämistä osittain muuhun kuin itse tiedonsiirtoon monimutkaista prosessia ja vaatien solmulaitteilta tämän huomioimista sovelluskoodissa. Näin päädyttiin luomaan oma protokolla, joka täyttäisi

opinnäytetyön vaatimukset, pitäen silti kiinni UDP:n tuomasta keveydestä, mutta lisäten TCP:n antamaa varmuutta (ks. kuvio 11).



Kuvio 11. Opinnäytetyön protokolla

Protokollan ylätunnisteen kooksi tuli 12 tavua, joka on sopiva välimalli UDP:n 8 tavun ja TCP:n 20 tavun ylätunnisteisiin nähden. 12 tavun ylätunniste sallii 39 tavun pakettikohtaisen hyötykuorman.

4.2 Tiedonsiirron osapuolien tunnistaminen

TCP:sta ja UDP:sta poiketen työssä ei tarvittu lähettävää eikä vastaanottavaa porttia, vaan lähettävä laite ja vastaanottava laiteryhmä. Näin ollen protokolla implementoi rakenteeltaan portteja vastaavat kentät ylätunnisteeseen, antaen kentille kuitenkin hieman eri merkityksen kuin mainituissa protokollissa.

Lähettävän laitteen tunnus muodostetaan 2 tavua pitkästä, laitteeseen tehtaalla syötetystä MAC-osoitteesta ja solmulaitteen komponenteista yhteenlaskettu uniikki arvo. Vastaanottavan laiteryhmän tunnus perustui yhtä pitkään heksadesimaaliarvoon. Arvo kuvastaa ryhmää, jonne vastaanottava laite voi rekisteröityä ja tällöin samaan laiteryhmään lähetetyt viestit näkyvät vain ryhmään rekisteröityneelle laitteelle.

4.3 Viestin tunnistaminen

Protokollaan tarvittiin keino tunnistaa, mihin viestiin saapuva paketti kuuluu, jos laite vastaanottaa useita paketteja samanaikaisesti. Tätä varten protokollaan luotiin 2 tavua pitkä viestitunnus (Message ID), joka generoidaan viestiä luotaessa.

Kahden tavun pituuden tukiessa vain 65 535:ta uniikkia arvoa, viestitunnuksen käyttöönottoa häiritsi ajatus mahdollisista konflikteista viestitunnusten välillä pidemmällä aikavälillä. Tunnuksen rinnalle harkittiin TCP/IP:sta tuttua Time to Live-arvoa. Arvo kuitenkin hylättiin, sillä:

- Paketteja ei välitetä uudelleen, ellei solmulaite saa muodostettua niistä kokonaista viestiä
- Valmiit viestit elävät rajatun kokoisessa FIFO-puskurissa, milloin vanhat viestit poistuvat itsestään puskurin täytyessä.

Konfliktien mahdollisuutta ehkäistäkseen entisestään viestitunnus suunniteltiin siten, ettei tunnus itsessään ole uniikki, vaan muodostaa uniikin yhdistelmän yhdessä lähettävän laitteen sekä vastaanottavan laiteryhmän tunnusten perusteella.

4.4 Virheenhallinta

4.4.1 Virheen tunnistaminen

Yksittäisen saapuvan paketin eheyden varmistamiseksi suoritetaan samanlaiset toimenpiteet kuin UDP- ja TCP-protokollissa laskemalla paketin tarkistussumma. Protokollat laskevat pakettien tarkistussummat yhden komplementilla, mutta opinnäytetyön protokolla implementoi tähän kevyen hajautusfunktion (engl. hash function), jonka arvo käännetään kahden tavun numeeriseksi arvoksi. Paketin ylätunnistetiedot siis sisältävät lähettävän laitteen laskeman tarkistussumman ja vastaanottava laite tekee vastaavat laskut verraten sitä ylätunnistetietojen tarkistussummaan. Jos arvot eivät täsmää, paketti luokitellaan virheelliseksi eikä sitä vastaanoteta.

UDP:n tavoin paketin mukana saapuu myös tieto ylätunnisteen ja hyötykuorman yhteispituudesta kentässä length yhden tavun pituisena numeerisena arvona, jolla varmistetaan myös ylätunnistetietojen saapuneen eheänä. Arvon on tarkoitus kuvata paketin kokonaispituutta tavun tarkkuudella, jolloin kenttä tukee paketin pituuden ilmaisuja 255 tavuun saakka.

Yksi UDP-protokollan ongelmista luotettavassa tiedonsiirrossa on vastaanotettavan datan järjestyksen ylläpito, sillä saapuvat paketit voivat päätyä eri järjestykseen kuin ne alun perin lähettäessä olivat. Vaikka viestien lähetys yhdellä solmulaitteella on synkroninen operaatio ja paketit lähetetään järjestyksessä, tästä tulee ongelma useiden laitteiden välittäessä samaan viestiin kuuluvia paketteja mesh-verkossa. Jotta saapuvat viestit olisivat mahdollisimman luotettavia eikä pakettien väärästä järjestyksestä johtuvaa epämuodostunutta dataa pääsisi syntymään, protokollaan implementoitiin TCP-protokollasta tuttu sekvenssinumerointi (engl. sequence number). Tällä numeroinnilla vastaanottava laite osasi jäsentää saapuvien pakettien hyötykuormat oikeaan järjestykseen valmista viestiä luotaessa.

Sekvenssinumeroinnille varattiin 2 tavua, jolloin valmis viesti voi olla jopa 65 535:n tavun, eli 1681 paketin pituinen.

4.4.2 Tilanhallinta

Jotta paketin tai viestin eheyden varmistuksen aikana kyettiin havaitsemaan jonkin paketin olevan viallinen tai puuttuvan, tuli protokollan kyetä pyytämään lähellä olevilta solmulaitteilta uutta pakettia sen tilalle. Tätä ei kuitenkaan kyetty toteuttamaan ilman jonkinlaista tilanhallintaa paketeille, joten protokollaan oli implementoitava TCP-protokollasta tutut lipputiedot (engl. flags). Kentän tarkoitus oli kuljettaa tietoa paketin tai viestin tilasta. Kentälle varattu pituus paketin ylätunnisteesta on yhden tavun pituinen. Protokolla implementoi seuraavat lipputiedot:

- BRO (Broadcast)
- REQ (Request)
- END.

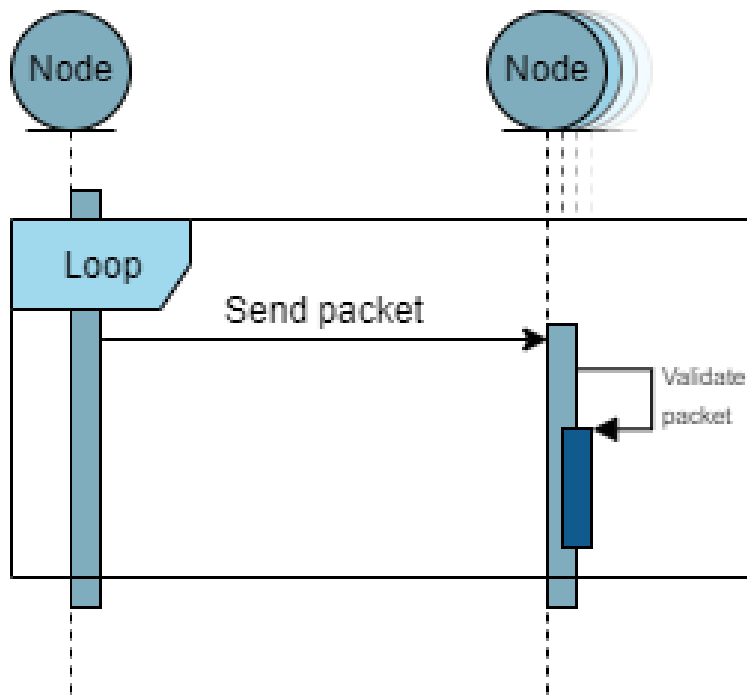
Broadcast, eli lähetyslippu, on oletusarvoinen lipputieto siitä, että paketti kulkee verkossa ja kantamalla olevien laitteiden tulee vastaanottaa se. Lipputietoa käytettiin osittain tunnistuksessa siitä, että paketti kuuluu opinnäytetyön protokollaan eikä muuhun verkossa kulkevaan liikenteeseen.

Request, eli pyyntölippu, liitetään vastaanottavan laitteen toimesta yhdessä sekvenssinumeron kanssa pakettiin, jonka viestiä vastaanottava solmulaite lähettää verkkoon. Pakettia käytettiin pyytämään uutta viestin sekvenssiä, jos solmulaite on todennut jonkin saapuneen paketin virheelliseksi tai jos viestin todetaan olevan epätäydellinen lähetyksen loputtua. Request-liputetun paketin vastaanottaneet laitteet luovat uuden Broadcast-paketin request-liputetun paketin sekvenssinumeron perusteella. Request-liputettu paketti ei sisällä hyötykuormaa.

END, eli lopetuslippu, kertoo viestin lähetyksen lopetuksesta ja viestin viimeisestä paketista. END-liputettu paketti jakaa Broadcast-paketin toiminnallisuuden mainitulla lisäinformaatiolla. Paketin saapuessa vastaanottavalle solmulaitteelle osaa laite aloittaa kokonaisen viestin eheyden varmistuksen.

4.5 Tiedonsiirto

Tiedonsiirtoa suunniteltaessa tavoite oli pysyä UDP:n tavoin yhteydettömässä protokollassa. Opinnäytetyön protokolla ei siis oletusarvoisesti toteuttanut minkäänlaista tilakonetta, eivätkä lähettävä tai vastaanottava laite tiedä toistensa olemassaolosta, sillä TCP-protokollasta tuttuja kuittausviestejä saapuneista paketeista ei lähetetty. Poikkeustapauksissa kyetään kuitenkin pyytämään uutta pakettia puuttuvan tai virheellisen tilalle. Kuvio 12 havainnollistaa tiedonsiirron periaatteen. Huomioitavaa on, että vastaanottavia solmulaitteita voi olla rajattomasti, kunhan ne ovat lähettävän laitteen lähetyskantaman sisäpuolella.



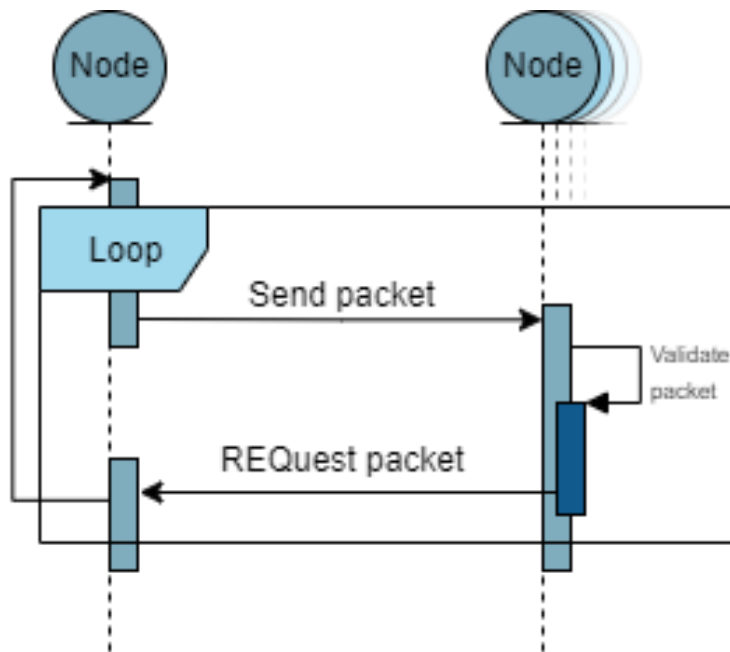
Kuvio 12. Tiedonsiirto kahden laitteen välillä

Jokaisen saapuvan paketin eheys tarkistetaan välittömästi vastaanoton jälkeen.

Saapuneesta paketista tarkistetaan seuraavat tiedot:

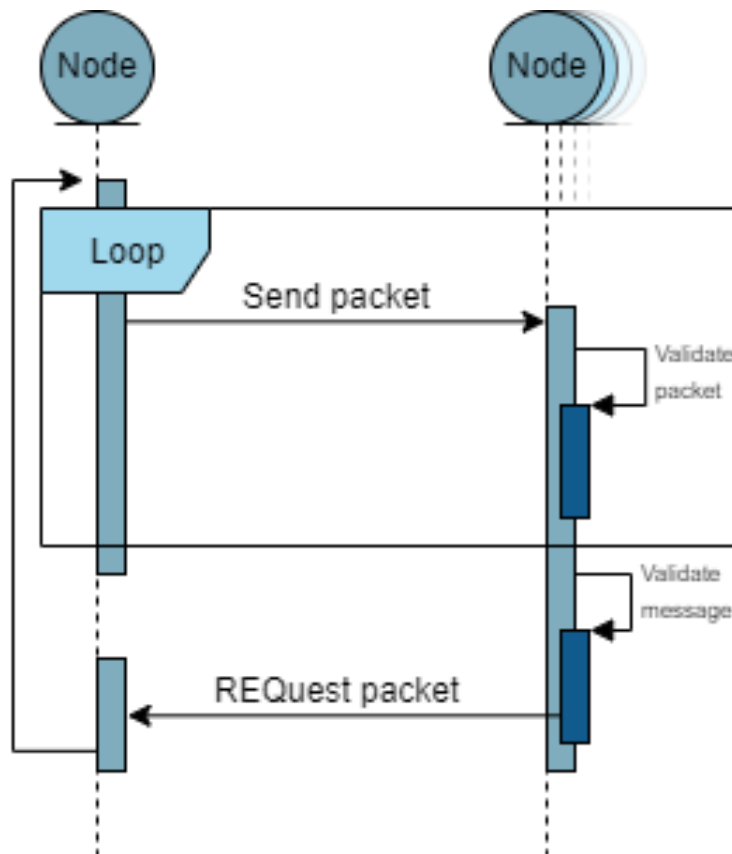
- Paketin tarkistussumma
- Paketin pituus.

Jos kumpikaan ehdoista jää täyttymättä, vastaanottava laite lähettää request-liputetun viestin virheellisestä paketista, jotta lähettävä laite osaa toimittaa uuden paketin viallisen tilalle. Viallinen paketti yksilöidään sekvenssitiedoilla (ks. kuvio 13).



Kuvio 13. Korvaavan paketin pyytäminen viallisen tilalle

Lähettävä solmulaite merkitsee toimitettavia paketteja luodessaan viimeisen paketin END-lipputiedolla kertoakseen vastaanottavalle laitteelle viestikokonaisuuden olevan valmis. Vastaanottava laite voi END-liputetun paketin vastaanotettuaan tarkistaa kokonaisen viestin eheyden järjestelemällä paketit sekvenssitietojen perusteella. Jos paketteja järjestellessä huomataan jonkin niistä puuttuvan, lähetetään siitä request-liputettu paketti puuttuvan paketin saamiseksi (ks. kuvio 14).



Kuvio 14. Puuttuvan paketin pyytäminen

Jos saapunut viesti viesti todetaan eheäksi ilman, että uusia paketteja tarvitsee pyytää, voidaan se lähettää eteenpäin pitkäaikaista säilöä tai prosessointia varten.

5 Kehitysympäristöt ja -välineet

5.1 Kehitysalustan valinta

Mikrokontrolleri on sulautettujen järjestelmien kehitystä varten luotu piirilevy, joka on pienikokoinen ja vähävirtainen, kuin kevyt tietokone. Mikrokontrolleri sisältää prosessorin, tallennusmuistin, ajonaikaisen muistin (RAM) ja input/output (I/O) -rajapinnan oheislaitteiden kanssa keskustelua varten (Rouse & Shea n.d.).

Mikrokontrollerit ovat usein tiettyyn tarkoitukseen, kuten robotiikkaan, mobiililaitteisiin, kodinelektroniikkaan tai Internet of Things-laitteisiin integroituja, missä tilaa ja virtaa on rajallisesti.

Mikrokontrollerit ovat kuitenkin pelkkään laskentaan suunnattuja piirejä, jotka eivät sisällä lainkaan oheislaitteita teknologioille kuten Bluetooth, Wi-Fi, LoRa jne. Näin ollen mikä tahansa tekninen kokeilu valitulla mikrokontrollerilla vaatisi suunnitelman ja toteutuksen piirikaaviosta ja tilattavista osista tarvitseineen, ellei mikrokontrollerille olisi olemassa olevaa kehitysalustaa. Kehitysalusta on piirilevy, joka integroi valitun mikrokontrollerin lisäksi käyttökohteen ja mikrokontrollerin kyvykkyyden mukaan valitun määrän oheislaitteita. Kehitysalusta on suunniteltu helpoksi lähtöpisteeksi valittuun mikroprosessoriin tutustumiseen (Weil 2012).

Työn prototyyppilaitteen toteuttamiseen valittiin TTGO LoRa32 V1.0 kehitysalusta sen tehokkaan, että monipuolisen Espressif ESP32 System on Chip -mikrokontrollerin ja työssä tarvittavan, kehitysalustaan sisäänrakennetun Semtech SX1276 LoRa lähetin-vastaanottimen vuoksi. System on chipilla (SoC) tarkoitetaan mikrokontrollerin kehittyneempää versiota, yhden piirin sisään rakennettua järjestelmää, joka sisältää kaikki tarvittavat komponentit itsenäiseen laskentakykyyn (Neagu 2017).

Prototyyppilaitteen lähdekoodi julkaistiin GitHub-palvelussa projektinimellä *SanLa-classic*, käyttäjän *Monni* repositoriossa. Lähdekoodi julkaistiin MIT-lisenssillä, joka antaa sen käyttäjälle kaikki oikeudet levittää ja muokata koodia yksityisesti ja kaupallisesti, kuitenkin ilman minkäänlaisia takuita. Repositorion osoite on:

<https://github.com/Monni/SanLa-classic>

5.2 Espressif ESP32

ESP32 on Espressif:n valmistama System on Chip -mikrokontrolleri. Mikrokontrolleri sisältää kaksi 240MHz Tensilica Xtensa-arkkitehtuuriin perustuvaa 32-bit LX6 mikroprosessoria. Mikrokontrollerin mikroprosessorit on jaettu kahteen eri tarkoitukseen: Protocol CPU ja Application CPU. Protocol CPU hoitaa mikrokontrollerin kaikkien oheislaitteiden toiminnan jättäen Application CPU:n sovelluskoodille. Työn vaatiman verrattain raskaan laskennan vuoksi on perusteltua

valita työhön mikrokontrolleri sovelluskoodille omistetulla, tehokkaalla mikroprosessorilla.

Mikrokontrollerissa on 8 KB sisäänrakennettua SRAM-muistia (Static Random Access Memory) sovellusten datan säilömiselle (ESP32 Series Datasheet 2019, luku 3.1.2). SRAM-muisti on transistoreilla toteutettu, haihtuvan muistin (engl. volatile memory), muistipiiri, joka säilyttää tietonsa ainoastaan niin kauan kuin laitteessa on virtaa (Pal n.d.), eikä näin ole kokonsa ja toteutustapansa vuoksi sopiva viestien pitkäaikaissäilölle.

Mikrokontrolleri kuitenkin tukee ulkoisia Flash-muistipiirejä jopa 16 MB kokoon saakka (ESP32 Series Datasheet 2019, luku 3.1.3). Flash-muisti on haihtumatonta muistia (engl. non-volatile memory), joka kykenee säilyttämään tallennetun datan, vaikka siihen yhdistetystä laitteesta katkaistaisiin virta (How Flash Memory Works – Advantages & Disadvantages 2018). Oletusarvoisesti ESP32-mikrokontrollerin mukana ei tule ulkoisia Flash-muistipiirejä, mutta useat kolmansien osapuolien toimittajat lisäävät nämä laitteisiinsa. TTGO LoRa32 V1.0 kehitysalusta valikoitui opinnäytetyöhön sisältämänsä 4 MB Flash-muistin vuoksi.

5.3 Semtech SX1276

SX1276 on Semtechin valmistama LoRa-yhteensopiva, 868 MHz taajuudella toimiva lähetin-vastaanotin tiedon lähettämiseen ja vastaanottamiseen pitkien matkojen päähän. Kuvion 15 esittämistä SX1276-moduulin ominaisuuksista työlle tärkeimmät ominaisuudet ovat korkea +20 dBm säteilyteho ja suuri -148 dBm herkkyys.

KEY PRODUCT FEATURES

- ◆ LoRa™ Modem
- ◆ 168 dB maximum link budget
- ◆ +20 dBm - 100 mW constant RF output vs. V supply
- ◆ +14 dBm high efficiency PA
- ◆ Programmable bit rate up to 300 kbps
- ◆ High sensitivity: down to -148 dBm
- ◆ Bullet-proof front end: IIP3 = -11 dBm
- ◆ Excellent blocking immunity
- ◆ Low RX current of 9.9 mA, 200 nA register retention
- ◆ Fully integrated synthesizer with a resolution of 61 Hz
- ◆ FSK, GFSK, MSK, GMSK, LoRa™ and OOK modulation
- ◆ Built-in bit synchronizer for clock recovery
- ◆ Preamble detection
- ◆ 127 dB Dynamic Range RSSI
- ◆ Automatic RF Sense and CAD with ultra-fast AFC
- ◆ Packet engine up to 256 bytes with CRC
- ◆ Built-in temperature sensor and low battery indicator

Kuvio 15. SX1276 tärkeimmät ominaisuudet (SX1276/77/78/79 - 137 MHz to 1020 MHz Low Power Long Range Transceiver 2019, 1).

Lähettimen säteilyteho voidaan vertailua varten kääntää milliwateiksi (mW) seuraavalla laskukaavalla (ks. kaava 5).

$$P_{(mW)} = 1mW * 10^{\left(\frac{P_{(dBm)}}{10}\right)} \quad (5)$$

Koska työn prototyypilaitteessa käytetty kehitysalusta toimitettiin isotrooppisella antennilla eikä ERP-laskutavan mukaisella puolialtodipoliantennilla, kaava ei suoraan tuota haluttua vastausta efektiivisestä säteilytehosta, vaan efektiivisestä isotrooppisesta säteilytehosta. Isotrooppinen antenni eroaa puolialtodipoliantennista toiminnallisuudeltaan siten, että se lähettää radioaaltoja tasaisella teholla kaikkiin suuntiin. Kun säteilytehoa suunnataan ja mitataan yhdestä suunnasta, puhutaan efektiivisestä isotrooppisesta säteilytehosta, lyhenteeltään EIRP (engl. Effective Isotropic Radiated Power) (Effective Isotropic Radiated Power (EIRP))

n.d.). Tätä säteilytehoa laskettaessa antennityypit erotellaan lyhenteillä dBi kuvaamaan isotrooppista antennia ja dBd kuvaamaan puoliaaltodipolian antennia.

Kun säteilytehon laskukaavaan (ks. kaava 5) lisättiin SX1276-lähettimen suurin säteilyteho +20 dBm ja antennivahvistus 2.1 dBi, yhteensä +22.1 dBm, saatiin lähettimen EIRP-arvo (ks. kaava 6).

$$162,181mW = 1mW * 10^{\left(\frac{22,1}{10}\right)} \quad (6)$$

Laskettu 162,181mW EIRP-arvo voidaan kääntää ERP-arvoksi kaavalla 7 (Guidelines for determining the effective radiated power (ERP) and equivalent isotropically radiated power (EIRP) of an RF transmitting system, luku 1.3).

$$ERP(W) = \frac{EIRP(W)}{1,64} \quad (7)$$

Tällöin ERP-arvoksi saatiin (ks. kaava 8).

$$98,891mW = \frac{162,181mW}{1,64} \quad (8)$$

Tuloksesta voitiin todeta moduulin lähes 100mW efektiivisen säteilytehon ylittävän Liikenne- ja viestintäviraston määräykset melkein nelinkertaisesti. Pysyäksemme määräysten rajoissa, prototyyppilaitteen tehoa tuli laskea kehityksen aikana. Prototyyppilaitteen vaatima säteilyteho voitiin laskea kaavalla 9.

$$25 mW = \frac{1mW * 10^{\left(\frac{x}{10}\right)}}{1,64} \quad (9)$$

Ratkaisemalla kaavan 9 yhtälöstä x saatiin laitteen suurimmaksi sallituksi säteilytehoarvoksi 16,1278 dBm. Arvosta vähentämällä antennin vahvistus 2,1 dBi voitiin prototyyppilaitteen tehoksi asettaa enimmillään 14 dBm. Laitteen puutteellisen dokumentaation vuoksi laskuissa ei huomioitu negatiivisia tekijöitä,

kuten liittimistä ja johdoista aiheutunutta tehon menetystä. Prototyypilaitteen oikea efektiivinen säteilyteho voi siten olla hieman laskettua alhaisempi.

SX1276 teknisestä esitteestä selvisi moduulin huolehtivan automaattisesti Liikenne- ja viestintäviraston määrittelemässä 1% suurimman sallitun toimintasuhteen rajoissa pysymisestä toimiessa ainoastaan suurimmalla +20 dBm säteilyteholla (SX1276/77/78/79 - 137 MHz to 1020 MHz Low Power Long Range Transceiver 2019, luku 5.4.3). Asetettaessa työssä käytettävä SX1276 lähetin-vastaanotin lähettämään 14 dBm teholla syntyi tarve ohjelmistollisesti toteutettavaan valvontaan määrättyissä toimintasuhteen rajoissa pysymisestä.

5.4 Ohjelmointikielet ja -työkalut

Kaikki sulautettujen järjestelmien ohjelmointi suoritettiin C++ -ohjelmointikielellä. Soveltuvuustutkimukseen tuotetussa koodissa keskityttiin toteutuksen mahdollisimman helppoon skaalautuvuuteen eri laitteiden välillä, joten kaikki laitespesifiset ulkoiset kirjastot jätettiin koodista pois. Esimerkiksi ESP32- ja Arduino-laitteisiin on paljon kyseiselle laitteelle ohjelmistokehitystä helpottavia kirjastoja, jotka eivät toimi hyvin yhteen keskenään ja sitovat sovelluksen toimimaan ainoastaan yhdessä laitetypissä.

Ohjelmointiympäristönä sulautettujen järjestelmien ohjelmoinnissa käytettiin Visual Studio Code -tekstieditoria. Visual Studio Code on kevyt avoimen lähdekoodin ohjelmisto, joka toimii useammalla järjestelmälustalla kuten Windowsilla ja Linuxilla, tukee laajaa valikoimaa liitännäisiä ja sisältää useita sovelluskehitystä helpottavia työkaluja, kuten koodin automaattisen täydentämisen. Yhdeksi projektille tärkeimmistä liitännäisistä valikoitui sulautettujen järjestelmien ohjelmointia tukeva PlatformIO. Ympäristön kuvataan keskittävän laitekehityksen ja niissä tarvittavat kirjastot yhteen paikkaan tehden siitä houkuttavan kehitysympäristön.

Different microcontrollers normally have different developing tools. For instance Arduino rely on Arduino IDE. Few more advanced users set up different graphical interfaces like Eclipse for better project management. Sometimes it may be hard to keep up with different microcontrollers and tools. You probably thought that single unified development tool could be great. Well this is what PlatformIO open source ecosystem is for (PlatformIO is an open source ecosystem for IoT development n.d., Press about PlatformIO).

Soveltuvuustutkimuksessa toteutetun prototyypilaitteen ja sitä ohjaavan tietokoneen välinen graafinen käyttöliittymä kirjoitettiin Python 3.7 - ohjelmointikielellä tkinter-kirjastoa käyttäen. Tkinter on graafisten käyttöliittymien toteutukseen suunniteltu kirjasto, joka kuuluu Pythonin standardikirjastoihin.

6 Verkkoprotokollan toteutus

6.1 Viesti

Tiedonsiirron toteutus aloitettiin toteuttamalla luokka, jonka tarkoitus oli kyetä käsittelemään verkossa liikkuvia kokonaisia viestikokonaisuuksia.

Viesti on rakenne, joka sisältää valmiin loppukäyttäjän tai -laitteen luoman hyötykuorman lähetettäväksi yhdeltä päätelaitteelta toisille. Jotta päätelaitteen rajattua muistia käytettäisiin mahdollisimman tehokkaasti, viesti tallentaa ainoastaan välttämättömimmät tiedot sen verkossa kuljettamiselle. Sellaiset tiedot, kuten hyötykuorman tarkistussumma, voidaan aina laskea pakettikohtaisesti tarvittaessa eikä näitä ollut syytä tallentaa osaksi viestiä.

Viesti koostettiin ohjelmistollisesti yhdestä luokasta, joka voitiin jakaa kahteen osaan: Ylätunniste (engl. header) ja runko (engl. body) (ks. kuvio 16).

```

class SanlaMessagePackage {
    MessageHeader header;
    Payload_t body;

public:
    SanlaMessagePackage(MessageId_t, SenderId_t, PayloadChecksum_t, RecipientId_t, Payload_t);
    SanlaMessagePackage(MessageHeader, Payload_t);

    ~SanlaMessagePackage(){};

    uint16_t GetPackageLength();
    MessageHeader& GetPackageHeader();
    Payload_t& GetPackageBody();
    PayloadChecksum_t& GetPackagePayloadChks();
};

```

Kuvio 16. Viestin rakenne

6.1.1 Viestin ylätunniste

Viestin ylätunniste rakennettiin kolmesta elementistä, jotka yhdessä tekivät viestistä uniikin (ks. kuvio 17).

```

struct MessageHeader {
    MessageId_t message_id;
    SenderId_t sender_id;
    RecipientId_t recipient_id;
};

```

Kuvio 17. Viestin header-elementti

message_id on yksilöity viestitunnus erottamaan viesti muista saman ryhmän viesteistä. Muuttuja sisältää UUID4 -tyyppisen arvon.

sender_id antaa tunnisteen viestin alkuperäisen lähettäjälaitteen tunnistamiseksi. Muuttuja generoidaan laitekohtaisesti.

recipient_id sisältää viestin lähettäjälaitteen määrittelemän ryhmätunnuksen viestin vastaanottavasta kohderyhmästä, jossa vain kohderyhmään rekisteröityneet laitteet pystyvät lukemaan lähetetyn viestin.

6.1.2 Viestin runko

Viestin runko sisältää varsinaisen lähetettävän hyötykuorma kokonaisuudessaan. Runko koostettiin yhdestä char array -tyyppisestä merkkijonosta, jonka pituudesta tehtiin vapaasti konfiguroitava käytettävän solmulaitteen kyvykkyyden mukaan.

Opinnäytetyössä käytetyt laitteet eivät varsinaisesti määritelleet viestin pituudelle teknistä enimmäisrajaa, mutta työtä tehdessä leikiteltiin ajatuksella

Puolustusvoimien jo käytöstä poistaneen sanomalaitteen, Sanomalaite ”SANLA” M/90, pienoisversiosta ”SANLA Classicista”, jonka alkuperäisversiota kunnioittaen viestin enimmäispituus määriteltiin 2000 merkkiin alkuperäisen sanomalaitteen mukaisesti.

6.2 Paketti

Paketti on rakenne, joka sisältää kaikki tarvittavat tiedon viestin koostamiselle ja liikuttamiselle verkossa. Lähetettävä viesti voidaan jakaa yhteen tai useampaan pakettiin riippuen lähetettävän viestin hyötykuorman pituudesta. Paketti jaettiin viestin tavoin ylätunnisteeseen ja runkoon (ks. kuvio 18).

```
struct SanlaPacket {
    SanlaPacketHeader header;
    sanlapacket::Payload_t body;

    void copy_headers_from_message(MessageHeader, sanlamessage::Payload_t);
};
```

Kuvio 18. Paketin rakenne

6.2.1 Paketin ylätunniste

Paketin ylätunnisteessa kulkevat viestin ylätunnistetietojen (ks. luku 6.1.1) lisäksi sekvenssitiedot viestin rungon osasta viestin koostuessa useammasta paketista, pakettikohtainen rungon tarkistussumma ja lipputieto (engl. flag) siitä, minkä tyyppinen paketti sisältö on (ks. kuvio 19).

```
const Flags_t BRO {0x0}, END {0x1}, REQ {0x2};
```

You, a few seconds ago | 1 author (You)

```
struct SanlaPacketHeader {  
    Flags_t flags;  
    uint16_t message_id;  
    uint16_t sender_id;  
    uint16_t recipient_id;  
    uint8_t payload_length;  
    uint16_t payload_seq;  
    uint16_t payload_chks;  
};
```

Kuvio 19. Paketin ylätunniste

6.2.2 Paketin runko

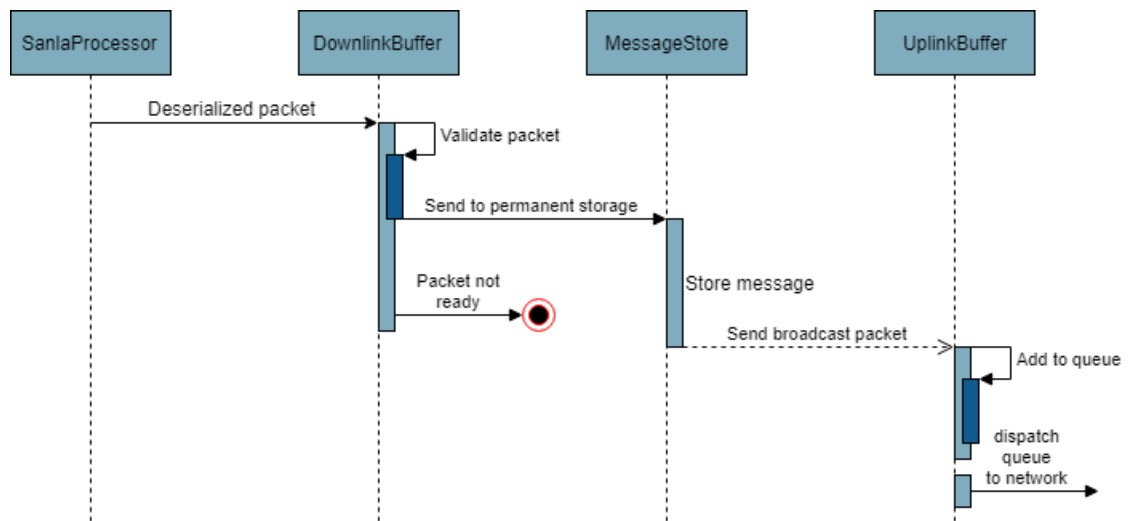
Paketin runko koostuu viestin tavoin pituudeltaan konfiguroitavasta viestijonosta, jonka pituus suhteutettiin LoRa-lähetin-vastaanottimeen määriteltyihin lähetysohjeisiin ja suosituksiin sallitusta pakettikoosta (ks. taulukko 4).

Määrittäessä opinnäytetyön protokollaa viestijonon pituudeksi, eli paketin hyötykuormaksi, syntyi 39 tavua. Tällöin jokainen yksittäisen paketin pituus pysyi määritellyssä 51 tavussa ja jokaisen 39 tavua ylittävän viestin hyötykuorma pilkottiin vastaavan pituisiksi pakettien hyötykuormiksi.

6.3 Tietoliikenne

6.3.1 Toimintaperiaate

Viestien prosessointia varten luotiin neljä luokkaa. Jokaisella luokalla oli erillinen toimintatarkoitus kokonaisuudessa, joka huolehti viestien vastaanotosta, tallennuksesta ja lähetyksestä (ks. kuvio 20).



Kuvio 20. Saapuvien pakettien prosessointi

6.3.2 MessageStore

Luokka, joka toimii saapuvien viestien pitkäaikaissäilönä. Säilöttävät viestit (ks. luku 6.1) tallennetaan solmulaitteen Flash -muistiin, jotta viestit säilyvät pysyvästi jopa solmulaitteen menettäessä virtansa. Tällöin viestit ovat jälleen saatavilla laitteen uudelleenkäynnistyksen jälkeen.

Toiminnallisuudeltaan MessageStore toteutettiin yksinkertaiseksi. Luokka sisältää funktiot ainoastaan viestin tallentamiselle ja noutamiselle sekä viestistä irrotettavan yksittäisen paketin noutamiselle.

Viestin yksittäisen paketin hakemisen tarkoitus oli mahdollistaa muiden laitteiden kysely puuttuvien pakettien perään. Funktio ottaa parametreina vastaan viestitunnuksen ja sekvenssitiedon siitä, mistä osasta viestiä hyötykuorma tulee pilkkoa.

6.3.3 UplinkBuffer

Puskuriluokka, joka huolehtii pakettien lähettämisestä verkkoon. Luokkaan toteutettiin yksityinen vektori, joka toimii FIFO-tyylisenä lähetyspuskurina verkkoon lähetettävälle paketeille. Vektorista puhuttaessa tarkoitetaan C++ standardikirjaston mukaista vector-containeria, joka toimii dynaamisena elementtijonona.

Luokka implementoi kaksi julkista metodia. Ensimmäinen metodi on pakettien lisäys puskuriiin. Puskurin koko oli konfiguroitavissa ja opinnäytetyön solmulaitteissa määritelty 50 paketin kokoiseksi. Puskurin pitäisi tyhjetä nopeasti, mutta verkossa mahdollisesti liikkuva suuri datamäärä voi täyttää tämän aiheuttaen pakettien menetyksiä. Puskurin ollessa täynnä sinne lisättäessä uutta lähetettävää pakettia, yritetään puskuria tyhjentää lähettämällä paketteja verkkoon ennen uuden lisäämistä jonoon. Jos puskuriiin ei vielä kukaan saatu tilaa, on lähetettävä paketti pudotettava liikenteestä. Protokollan virreehallinta osaa reagoida tähän jossain määrin ja vastaanottava solmulaite osaa pyytää uusia paketteja puuttuvien tilalle, mutta odottamattomia ongelmia voi syntyä ja puskurin kasvattamista on voitava harkita.

Toinen metodi on puskurin tyhjentäminen verkkoon päin. Metodia kutsutaan jokaisella solmulaitteen ajonaikaisella kierroksella (engl. loop) jolloin puskuri pyrkii lähettämään paketteja verkkoon järjestyksessä niin paljon kuin pystyy. Pakettia verkkoon lähettäessä kutsutaan LoRa-moduulin toiminnasta vastaavan kirjaston funktiota, ja onnistuneen lähetyksen jälkeen lähetetty paketti poistetaan puskurista. LoRa-moduuli ilmaisee lähetysvalmiutensa arvoilla 1 ja 0. Kertoessaan olevansa epävalmis lähetykseen puskurin tyhjennys lopetetaan ja pakettien lähetystä yritetään seuraavalla kierroksella uudelleen.

Luokan velvollisuuksiin kuuluisi myös lähetettävien pakettien valvonta siten, ettei Liikenne- ja viestintäviraston asettamaa määräystä radiolaitteen suurimmasta sallitusta toimintasuhteesta ylitetä. Valvonta kuitenkin rajattiin työn ulkopuolelle ja testausvaiheessa lähetettiin tarvittavan pieni määrä paketteja jaettuna eri testikertoihin, ettei rajoja ylitetty.

6.3.4 DownlinkBuffer

Puskuriluokka, johon saapuvien pakettien data säilötään väliaikaisesti, kunnes siitä kyetään luomaan valmis viesti. Laitteen asettamien muistirajoitteiden vuoksi sitä tuli käyttää mahdollisimman konservatiivisesti pakettien tallennuksessa. Ihanteellisen muistinkäytön vuoksi luokka ei voinut suoraan implementoida verkossa kulkevaa

pakettia (ks. luku 6.2), vaan saapuvasta paketista luotiin erillinen, niiden tilapäistä säilömistä varten suunniteltu DownlinkPacket.

DownlinkPacket on lähes paketin veroinen rakenne, mutta josta on riisuttu paketin tilatiedot ja suunniteltu säilömään kaikki seuraavat samaan viestiin kuuluvat pakettien hyötykuormat itseensä eikä yksittäisinä paketteina (ks. kuvio 21).

```
struct DownlinkPacket {
    uint16_t message_id;
    uint16_t sender_id;
    uint16_t recipient_id;
    PayloadBuffer_t payloadBuffer;
};
```

Kuvio 21. DownlinkPacketin rakenne

DownlinkBuffer-luokan perimmäinen idea oli toimia samantapaisena puskurina verkossa liikkuville paketeille kuten UplinkBufferilla (ks. luku 6.3.3), mutta vastaanottavassa päässä verkkoa. Jotta solmulaite kykenisi säilömään useiden viestien samanaikaisesti verkossa kulkevia paketteja, samaan viestiin kuuluvien pakettien hyötykuormien tallentaminen yhden DownlinkPacketin alle toteutettiin C++ standardikirjaston mukaisella map-containerilla, jonne arvot tallennetaan avain- ja arvopareina. Mapin hyöty vektoriin nähden on tuki hakea arvoja avaimen perusteella, missä vektorissa arvoja voidaan ainoastaan iteroida. DownlinkPacketin implementoimalle mapille annettiin kaksi avainta ja arvoksi saapuvan paketin hyötykuorma (ks. kuvio 22). Avaimina käytettiin saapuneen paketin sekvenssi- ja lipputietoja myöhemmin suoritettavaa valmistuvan viestin varmennusta varten.

```
using PayloadBuffer_t = std::map<std::pair<PayloadSeq_t, Flags_t>, std::string>;
```

Kuvio 22. DownlinkPacketin hyötykuormapuskuri

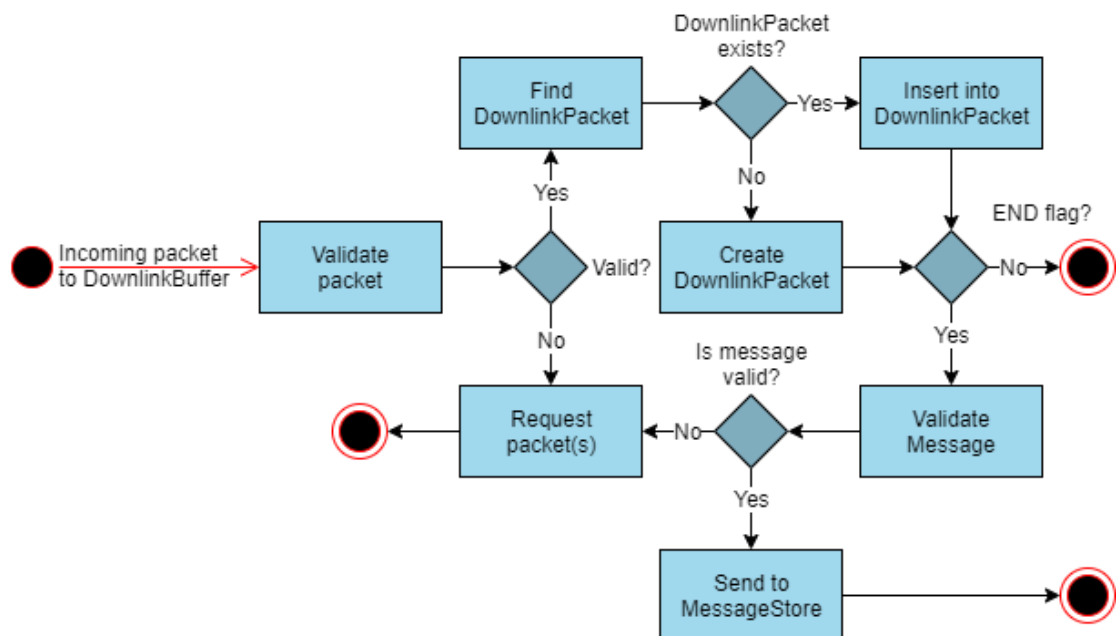
DownlinkBufferin toimintaperiaate oli se, että jos saapuva paketti kuuluu viestiin joka saa vasta ensimmäisen pakettinsa, käännettiin saapuneesta paketista DownlinkPacket ja tallennettiin se mapiin, joka säilöo DownlinkPacketeja

viestitunnuksen perusteella (ks. kuvio 23). Jos viestille oli jo olemassa oleva yksittäinen tai useampi paketti, haettiin viestitunnuksen (message_id) perusteella olemassaolevaa DownlinkPacketia ja lisättiin hyötykuorma DownlinkPacketin hyötykuormavektoriin.

```
using DownlinkPacketMap = std::map<MessageId_t, DownlinkPacket*>;
```

Kuvio 23. DownlinkBufferin DownlinkPacket-puskuri

Kun luokan instanssi ottaa paketin vastaan, ensimmäisenä varmistetaan saapuneen paketin eheys, joka toteutettiin kahdella eri varmistuksella. Kun saapunut paketti on tallennettu uudessa muodossaan, varmistetaan voiko DownlinkPacketin tiedoista luoda kokonaisen viestin. Varmistus toteutettiin iteroimalla jokaista vektorissa olevaa hyötykuormaa tämän sekvenssietoihin. Jos paketin todetaan olevan valmis, lähetetään se MessageStorelle pitkäaikaissäilöttäväksi kokonaisena viestinä ja poistetaan DownlinkPacket DownlinkBufferista (ks. kuvio 24).



Kuvio 24. DownlinkBufferin toimintaperiaate

Jos kummankaan varmistuksen aikana paketeissa huomattiin virheitä tai puuttuvia paketteja estäen valmiin viestin luomisen, pyydetään request-paketilla muilta

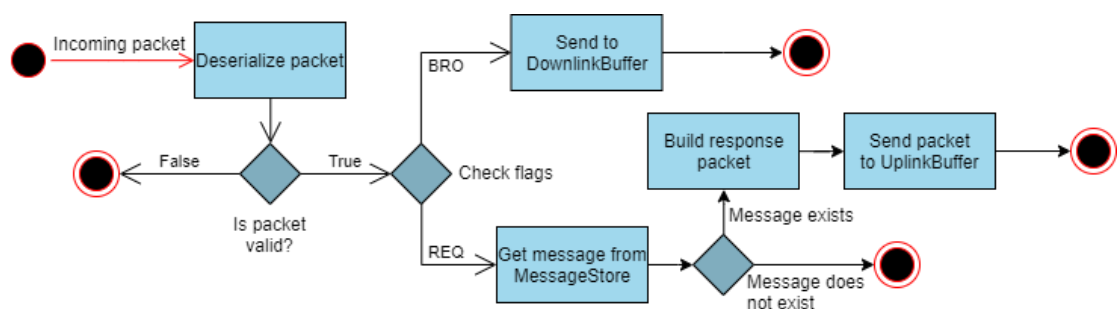
kantamalla olevilta laitteilta uusia paketteja täyttämään DownlinkPacketin epäkohdat.

6.3.5 SanlaProcessor

SanlaProcessor-luokka määriteltiin singleton instanssiksi, joka toimii ainoana rajapintana tietoliikenneprotokollan ja sitä käyttävän sovelluksen välissä. Luokka sisältää pakettien ja viestien liikuttamiseen vaadittavan logiikan ja luokka on vastuussa siitä, mihin saapunut paketti tai viesti tulee seuraavaksi lähettää prosessoitavaksi. Luokan tehtävä on toimia yhteisenä komponenttina kaikille toteutetuille luokille, joka myös sisältää suurimman osan laitteen logiikasta, sekä implementoi ja toimii ainoana rajapintana edellä mainittuihin luokkiin.

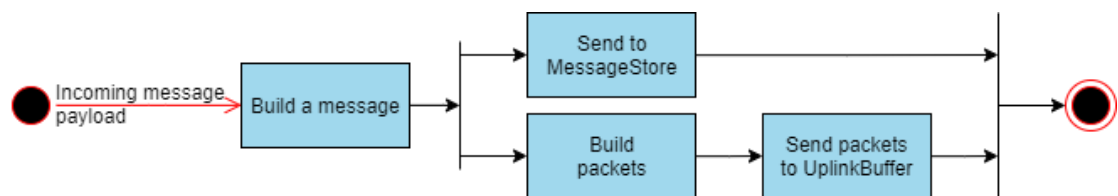
Luokan instanssi ottaa vastaan verkosta saapuvat, serialisoidut, viestijonot ja deserialisoi niistä laitteen käyttämiä paketteja (ks. luku 6.2). Kun paketti on luotu, tulee siitä selvittää lipputiedot ja niiden perusteella päätellä paketille haluttu toiminto.

Paketin saapuessa BRO -lipulla, voidaan paketin päätellä olevan osa aktiivista lähetystä, jolloin saapunut paketti osataan siirtää eteenpäin DownlinkBufferille tarkempaa varmennusta ja säilöntää varten. Mikäli saapuva paketti on REQ - liputettu, se on jonkin muun verkkoon kiinnittyneen laitteen paketti, jolla pyydetään paketin vastaanottaneelta laitteelta uutta pakettia jonkin puuttuvan tai viallisen tilalle. Tällöin paketin viestitunnuksella ja sekvenssitiedolla haetaan uutta pakettia MessageStoresta ja lähetetään UplinkBufferille, jos sellainen löytyi (ks. kuvio 25).



Kuvio 25. Saapuvan paketin vastaanotto

Luokan vastuulla on myös käyttäjän syötteen, tai muun lähetettävän datan, rakentamisesta verkkoonlähetykelpoiseksi ja sen säilömisestä. Luokka tarjoaa tähän rajapinnan, joka ottaa vastaan lähetettävän datan. Datasta rakennetaan kokonainen viesti (ks. luku 6.1) ylätunnistetietoineen ja lähetetään viesti MessageStorelle pitkäaikaissäilöttäväksi, kuten myös pilkotaan viestistä verkkoon sopivia paketteja, jotka siirretään eteenpäin UplinkBufferille lähetettäväksi (ks. kuvio 26).

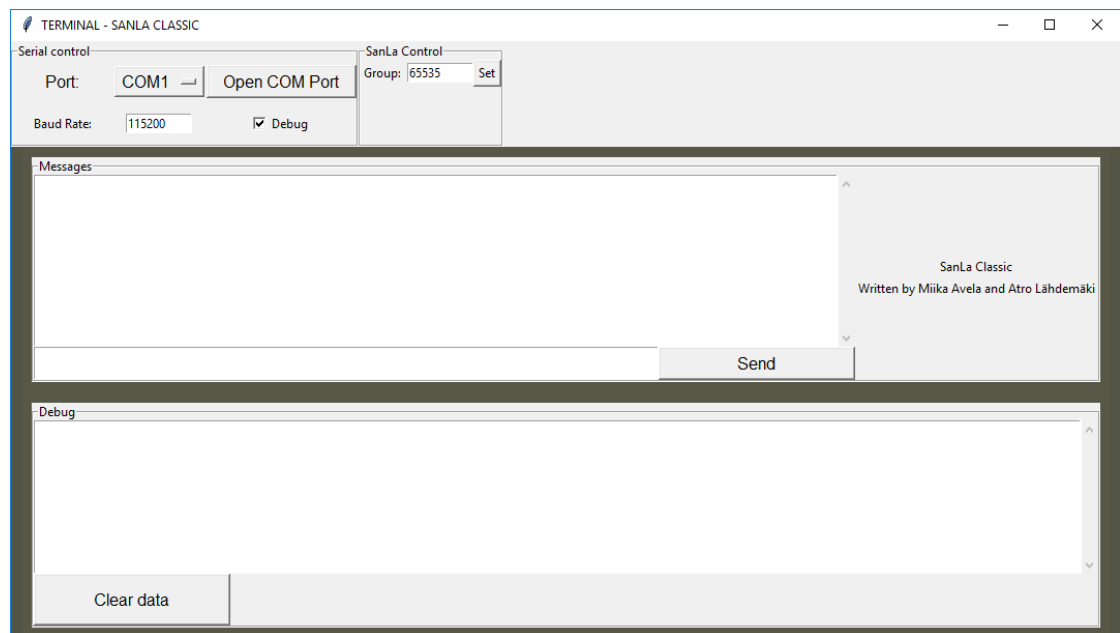


Kuvio 26. Saapuvan hyötykuorman vastaanotto

7 Testitapaukset ja tulokset

Opinnäytetyön soveltuvuustutkimuksessa toteutetun prototyyppilaitteen testaamiseen valittiin Jyväsjärven ympäristö Jyväskylässä. Alue tarjoaa pitkillä välimatkoilla sekä mahdollisuuksilla suoriin ja epäsuoriin yhteyksiin hyvän mahdollisuuden käyttötapauksissa (ks. luvut 2.3.1 ja 2.3.2) määriteltyjen skenaarioiden testaukseen.

Jotta skenaarioita voitiin testata helposti ja käyttäjäystävällisesti, soveltuvuustutkimuksessa käytettyjen laitteiden ympärille rakennettiin tietokonesovellus, graafinen käyttöliittymä, jolla kyettiin skenaarioiden vaatimusten mukaisesti muuttamaan solmulaitteen laiteryhmää ja lähettämään vaaditun mittaisia viestejä (ks. kuvio 27). Käyttöliittymä mahdollisti yhteyden solmulaitteeseen sen sarjaporttia pitkin, kun laite yhdistettiin USB-kaapelilla Windows, Linux tai Mac - tietokoneeseen ja sillä pystyi myös lukemaan laitteen sisäistä lokikirjaa protokollan sisäisistä tapahtumista.



Kuvio 27. Testikäyttöön suunniteltu käyttöliittymä

Ennen testiskenaarioiden aloittamista haluttiin varmistua yhteyden toimivuudesta kaksisuuntaisesti kahden solmulaitteen välillä. Sitä varten solmulaitteelta A lähetettiin yksinkertainen ”Hello world”-viesti solmulaitteelle B (ks. taulukko 5).

Taulukko 5. Yhteystestin viesti

Message	Length
Hello world!	12

Lähetetyn viesti tuli mahtua pituutensa vuoksi yhteen pakettiin, joka voitiin todeta lähetetyksi halutunlaisesti (ks. kuvio 28).

2019-10-17 21:52:52.631189 -- Sending packet for message 23395, sequence 0

Kuvio 28. Yhteystestin lähetetty paketti

Solmulaite B otti paketin vastaan, jonka jälkeen solmulaite lähetti saapuneen viestin verkossa eteenpäin automaattisesti (ks. kuvio 29), vaikkei verkossa ollut muita laitteita.

```
2019-10-17 21:52:54.546023 -- Received packet for message 23395, sequence 0
2019-10-17 21:52:54.548971 -- messaging::ActionsE::RECEIVE
2019-10-17 21:52:54.554021 -- DownlinkBuffer::StorePacket -- Creating new DownlinkPacket.
2019-10-17 21:52:54.557027 -- DownlinkBuffer::SendMessageToStore
2019-10-17 21:52:54.560040 -- SanlaProcessor::HandleMessage
2019-10-17 21:52:54.564974 -- LoRaModule::onPacket -- Processing incoming packet done!
2019-10-17 21:52:54.569000 -- Sending packet for message 23395, sequence 0
```

Kuvio 29. Yhteystestin paketin vastaanotto ja lähetys

Solmulaitteen A lokista voidaan myös havaita solmulaitteen vastaanottaneen juuri lähettämänsä viestin alle viisi sekuntia myöhemmin, mutta pudottaneen sen käsittelystä, sillä kyseinen viesti löytyi jo laitteen muistista (ks. kuvio 30).

```
2019-10-17 21:52:57.179026 -- DownlinkBuffer::ReceivePacket -- Message exists in store. Dropping packet.
```

Kuvio 30. Yhteystestissä vastaanotetun paketin pudotus verkosta

Sama testi toistettiin myös toiseen suuntaan lähettämällä vastaava yhden paketin mittainen viesti solmulaitteelta B solmulaitteelle A (ks. taulukko 6).

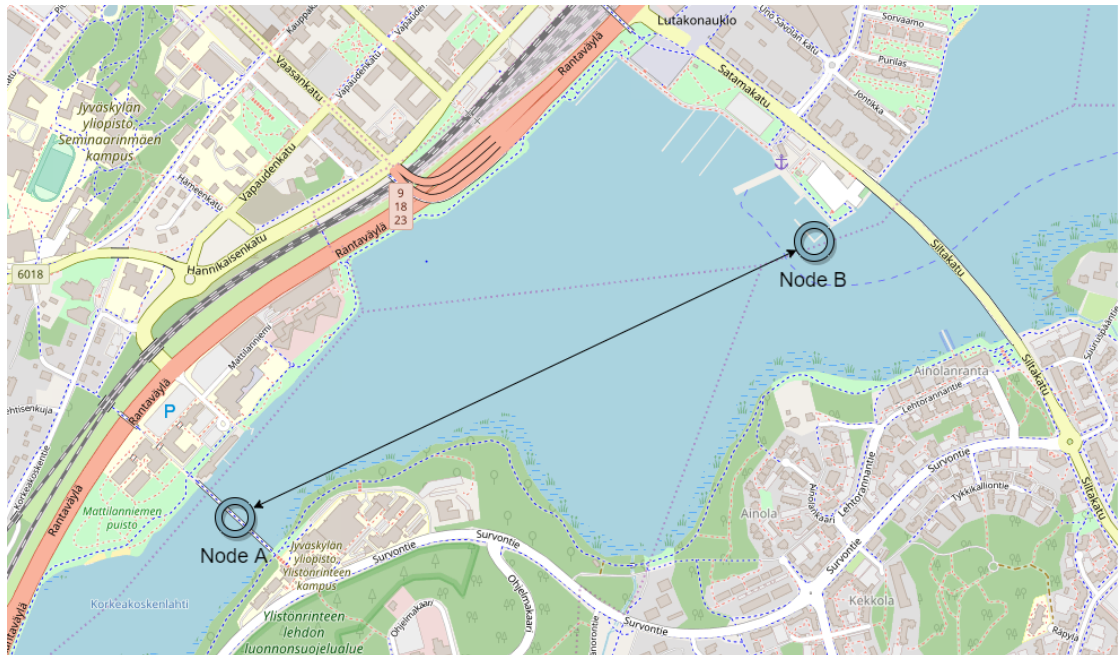
Taulukko 6. Yhteystestin vastausviesti

Message	Length
Hello from the other side.	26

solmulaite A:n vastaanotettua viesti onnistuneesti (ks. liite 1), voitiin kaksisuuntaisen yhteyden todeta toimivan solmulaitteiden välillä ja siirtyä määriteltyjen käyttöskenaarioiden testaamiseen (ks. luvut 2.3.1 ja 2.3.2). Koko yhteystestin kulku on nähtävissä liitteistä 1 ja 2.

7.1 Skenaario A: Viestin suora lähetys laitteelta toiselle.

Kahden solmulaitteen väliseen tiedonsiirtoon valittiin etäisyydeltään vaatimaton, noin 1.30km pituinen välimatka missä laitteilla oli suora näköyhteys toisiinsa, kuten kuvio 31 esittää.



Kuvio 31. Skenaarion A laiteasetelma

Skenaarion yhtenä esiehtona oli, että molempien solmulaitteiden piti olla rekisteröityneenä samaan laiteryhmään. Liitteistä 3 ja 4 voidaan todeta, että molemmat laitteet olivat samassa laiteryhmassä 65535.

Toisena esiehtona määrättiin, että lähetettävän viestin piti olla sellainen, ettei se pituutensa vuoksi mahdu lähetettäväksi yhdessä paketissa (ks. luku 2.3.2), joten testi toteutettiin lähettämällä 67 merkkiä pitkä viesti solmulaitteelta A solmulaitteelle B (ks. taulukko 7).

Taulukko 7. Skenaarion A testiviesti

Message	Length
This test message will be split into two parts during transmission.	67

Lähetetty viesti voitiin todeta lähteneen halutusti kahdessa paketissa (ks. kuvio 32).

```
2019-11-03 12:48:26.164189 -- 0: This test message will be split into two parts during transmission.
2019-11-03 12:48:26.164531 --
2019-11-03 12:48:26.167637 -- Sending packet for message 63420, sequence 0
2019-11-03 12:48:28.399954 -- Sending packet for message 63420, sequence 38
```

Kuvio 32. Skenaariossa A lähetetyt paketit

Solmulaitteen B vastaanottaessa paketteja, ensimmäinen paketti oli osa solmulaitteelle kokonaan uutta viestiä, jolloin solmulaite loi muistiinsa uuden DownlinkPacketin. Noin kaksi sekuntia myöhemmin vastaanotettiin jälkimmäinen viestiin kuuluva paketti, jolloin laite löysi viestin olemassaolevan DownlinkPacketin ja lisäsi saapuneen paketin hyötykuorman DownlinkPacketiin. Lopuksi laite varmisti saapuneen viestin eheyden ja asetti kokonaisen viestin nähtäväksi käyttöliittymän Messages-paneeliin (ks. kuvio 33).

```
2019-11-03 12:48:28.607126 -- LoRaModule::onPacket
2019-11-03 12:48:28.611129 -- Received packet for message 63420, sequence 0
2019-11-03 12:48:28.614132 -- messaging::ActionsE::RECEIVE
2019-11-03 12:48:28.619136 -- DownlinkBuffer::StorePacket -- Creating new DownlinkPacket.
2019-11-03 12:48:28.624141 -- LoRaModule::onPacket -- Processing incoming packet done!
2019-11-03 12:48:30.847217 -- LoRaModule::onPacket
2019-11-03 12:48:30.851221 -- Received packet for message 63420, sequence 38
2019-11-03 12:48:30.853223 -- messaging::ActionsE::RECEIVE
2019-11-03 12:48:30.859228 -- DownlinkBuffer::StorePacket -- Existing DownlinkPacket found.
2019-11-03 12:48:30.865233 -- DownlinkBuffer::StorePacket -- DownlinkPacket ready to MessageStore.
2019-11-03 12:48:30.868236 -- DownlinkBuffer::SendMessageToStore
2019-11-03 12:48:30.871239 -- SanlaProcessor::HandleMessage
2019-11-03 12:48:30.878245 -- 62612: This test message will be split into two parts during transmission.
```

Kuvio 33. Skenaariossa A lähetettyjen pakettien vastaanotto ja esitys

Kun viesti oli vastaanotettu ja laitettu esille, solmulaite B lähetti sen verkossa eteenpäin muille kantamalla oleville solmulaitteille pitäen viestin kahdessa paketissa (ks. kuvio 34).

```
2019-11-03 12:48:30.887253 -- Sending packet for message 63420, sequence 0
2019-11-03 12:48:33.118999 -- Sending packet for message 63420, sequence 38
```

Kuvio 34. Skenaariossa A vastaanotettujen pakettien välitys

Ainoa laitteen B kantamalla oleva solmulaite oli kuitenkin laite A, jonka muistissa välitettävä viesti oli, laitteen A ollessa välitettävän viestin alkuperäinen lähettäjä. Solmulaite A vastaanotti molemmat paketit ja pudotti ne verkosta onnistuneesti (ks. kuvio 35).

```
2019-11-03 12:48:32.959731 -- LoRaModule::onPacket
2019-11-03 12:48:32.963724 -- Received packet for message 63420, sequence 0
2019-11-03 12:48:32.966693 -- messaging::ActionsE::RECEIVE
2019-11-03 12:48:32.973552 -- DownlinkBuffer::ReceivePacket -- Message exists in store. Dropping packet.
2019-11-03 12:48:32.978667 -- LoRaModule::onPacket -- Processing incoming packet done!
2019-11-03 12:48:35.199721 -- LoRaModule::onPacket
2019-11-03 12:48:35.203623 -- Received packet for message 63420, sequence 38
2019-11-03 12:48:35.205723 -- messaging::ActionsE::RECEIVE
2019-11-03 12:48:35.212671 -- DownlinkBuffer::ReceivePacket -- Message exists in store. Dropping packet.
2019-11-03 12:48:35.217574 -- LoRaModule::onPacket -- Processing incoming packet done!
```

Kuvio 35. Skenaariossa A välitettyjen pakettien pudotus verkosta

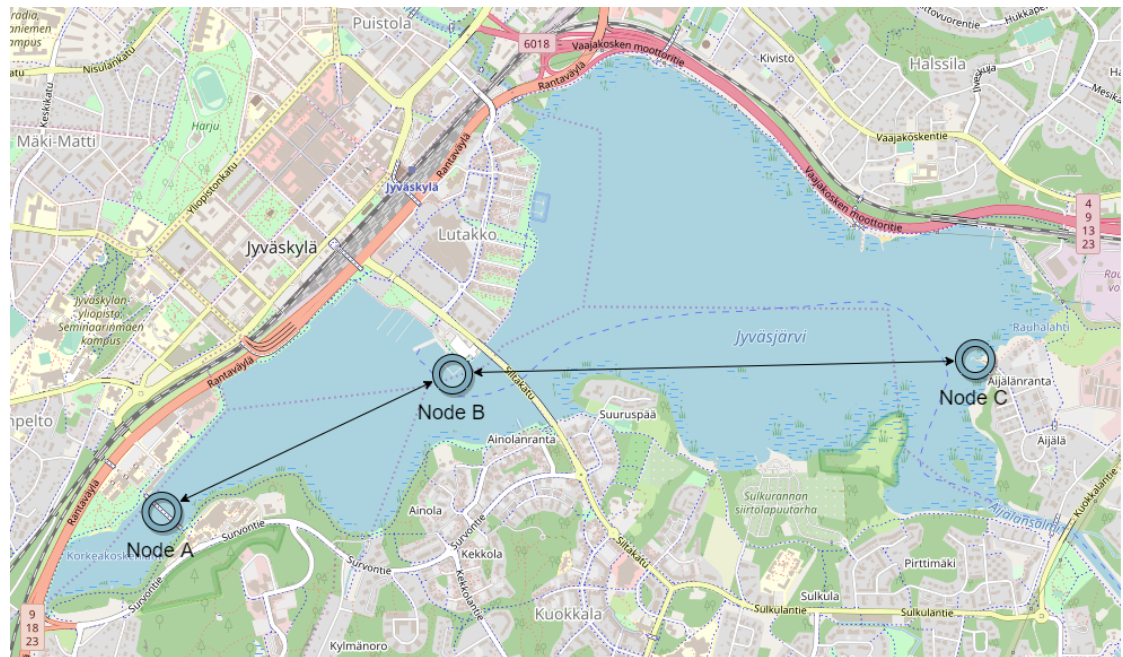
Testiskenaario täytti skenaarion A esiehdot ja vastasi vaadittua lopputilaa. Näin ollen testiskenaario vastasi tutkimuskysymykseen 1, jättäen auki kysymyksen mesh-topologisuudesta (ks. luku 2.2). Skenaarion A täydellinen kulku on nähtävillä liitteissä 3 ja 4.

7.2 Skenaario B: Viestin välillinen lähetys laitteelta toiselle.

Solmulaitteiden välillisen tiedonsiirron testausta varten skenaarion B (ks. luku 2.3.2) mukaisesti laiteasetelmaan lisättiin uusi solmulaite C, joka oli tässä testissä uusi vastaanottava solmulaite B:n sijaan (ks. kuvio 36).

Etäisyys solmulaitteiden B ja C välillä kasvatettiin noin 2 kilometriin, missä solmulaitteiden välillä oli vain osittainen näköyhteys. Lähettävän solmulaitteen A ja vastaanottavan solmulaitteen C välille tuli näin 3.30, linnunteitse 3.26, kilometrin näköyhteydetön välimatka. Teoriassa solmulaitteet A ja C olisivat voineet

kommunikoida tämän matkan suorasti keskenään, mutta se olisi myös vaatinut suoran näköyhteyden solmulaitteiden välille.



Kuvio 36. Skenaarion B laiteasetelma

Vastataksaan tutkimuskysymykseen eri laiteryhmiä välisestä kommunikaatiosta mesh-topologisuuden lisäksi, skenaarion testaus aloitettiin rekisteröimällä verkon solmulaitteet ryhmittäin. Toistensa kantaman ulkopuolella olevat laitteet A ja C rekisteröitiin omaan ryhmäänsä, ryhmään 255, kun molempien laitteiden kantamalla sijaitseva laite B sai ryhmänsä arvokseen 123 (ks. liitteet 5, 6 ja 7).

Testi aloitettiin lähettämällä 63 merkkiä pitkä viesti solmulaitteelta A (ks. taulukko 8).

Taulukko 8. Skenaarion B testiviesti

Message	Length
This message is not short, so it needs to be sent in two parts.	63

Pituutensa takia testiviesti pilkottiin ja lähetettiin kahdessa paketissa (ks. kuvio 37).

```

2019-10-17 22:00:32.365351 -- 0: This message is not short, so it needs to be sent in two parts.
2019-10-17 22:00:32.365351 --
2019-10-17 22:00:32.365351 -- Sending packet for message 57758, sequence 0
2019-10-17 22:00:34.584558 -- Sending packet for message 57758, sequence 38

```

Kuvio 37. Skenaariossa B lähetetyt paketit

Solmulaite B sai vastaanotettua molemmat paketit noin neljä sekuntia sen jälkeen, kun solmulaite A aloitti pakettien lähetyksen, kykenemättä näyttämään saapunutta viestiä, sillä solmulaite B ei määritellysti kuulunut samaan ryhmään solmulaitteen A kanssa (ks. kuvio 38).

```

2019-10-17 22:00:34.202658 -- DownlinkBuffer::StorePacket -- Creating new DownlinkPacket.
2019-10-17 22:00:34.207648 -- LoRaModule::onPacket -- Processing incoming packet done!
2019-10-17 22:00:36.430208 -- LoRaModule::onPacket
2019-10-17 22:00:36.434128 -- Received packet for message 57758, sequence 38
2019-10-17 22:00:36.437146 -- messaging::ActionsE::RECEIVE
2019-10-17 22:00:36.442116 -- DownlinkBuffer::StorePacket -- Existing DownlinkPacket found.
2019-10-17 22:00:36.449070 -- DownlinkBuffer::StorePacket -- DownlinkPacket ready to MessageStore.
2019-10-17 22:00:36.452068 -- DownlinkBuffer::SendMessageToStore
2019-10-17 22:00:36.454201 -- SanlaProcessor::HandleMessage
2019-10-17 22:00:36.459093 -- LoRaModule::onPacket -- Processing incoming packet done!
2019-10-17 22:00:36.463139 -- Sending packet for message 57758, sequence 0
2019-10-17 22:00:38.695495 -- Sending packet for message 57758, sequence 38

```

Kuvio 38. Skenaariossa B välitettävien pakettien vastaanotto ja lähetys

Pakettien vastaanoton jälkeen solmulaite B ryhtyi välittämään saapuneita paketteja muille lähetyksentamalla oleville laitteille pitäen lähtevät paketit muuttumattomina (ks. kuvio 38). Solmulaite C otti lähetetyt paketit vastaan samoin, kuten solmulaite B otti alkuperäiset paketit vastaan solmulaitteelta A ja kykeni esittämään niistä rakennetun viestin (ks. kuvio 39).

```

2019-10-17 22:00:38.754714 -- LoRaModule::onPacket
2019-10-17 22:00:38.758716 -- Received packet for message 57758, sequence 0
2019-10-17 22:00:38.761710 -- messaging::ActionsE::RECEIVE
2019-10-17 22:00:38.766724 -- DownlinkBuffer::StorePacket -- Creating new DownlinkPacket.
2019-10-17 22:00:38.771719 -- LoRaModule::onPacket -- Processing incoming packet done!
2019-10-17 22:00:40.993783 -- LoRaModule::onPacket
2019-10-17 22:00:40.997787 -- Received packet for message 57758, sequence 38
2019-10-17 22:00:41.000789 -- messaging::ActionsE::RECEIVE
2019-10-17 22:00:41.005793 -- DownlinkBuffer::StorePacket -- Existing DownlinkPacket found.
2019-10-17 22:00:41.011800 -- DownlinkBuffer::StorePacket -- DownlinkPacket ready to MessageStore.
2019-10-17 22:00:41.014802 -- DownlinkBuffer::SendMessageToStore
2019-10-17 22:00:41.017805 -- SanlaProcessor::HandleMessage
2019-10-17 22:00:41.023810 -- 128: This message is not short, so it needs to be sent in two parts.

```

Kuvio 39. Skenaariossa B välitettyjen pakettien vastaanotto ja esitys

Testiskenaarion kulku vastasi skenaarion B määrittelemää pääonnistumisskenaariota ja toteutti vaaditun lopputilan. Näin ollen testiskenario vastasi tutkimuskysymyksiin 1 ja 3 (ks. luku 2.2). Skenaarion B täydellinen kulku on nähtävissä liitteissä 5, 6 ja 7.

7.3 Poikkeusskenario: Paketti on jäänyt saapumatta perille.

Tiedonsiirron luotettavuutta testattiin erikseen muokkaamalla lähetettävää laitetta siten, että monivaiheisia viestejä lähettäessä laite tiputtaa aina jonkin yksittäisen paketin pois lähetyksestä. Toteutettu testiskenario vastasi skenaarioille A ja B mahdollisia poikkeustapauksia, joissa lähellä olevilta laitteilta voidaan pyytää uutta pakettia puuttuvan tilalle.

Testi aloitettiin lähettämällä laitteelta A sellainen viesti laitteelle B, että se tuli pituutensa puolesta pilkkoa neljään erikseen lähetettyyn pakettiin (ks. taulukko 9).

Taulukko 9. Poikkeusskenaarion testiviesti

Message	Length
This message contains 119 characters so it cannot be sent in a single transmission, but must be split across 4 packets.	119

Järjestyksessä jonon paketeista toinen, sekvenssi 38, pudotettiin pois ja voitiin todeta, ettei kyseisen sekvenssin pakettia lähetetty pakettien lähetyksen yhteydessä (ks. kuvio 40).

```
2019-10-12 22:54:36.836251 -- 0: This message contains 119 characters so it cannot be sent in a single transmission, but must be split across 4 packets.
2019-10-12 22:54:36.836251 --
2019-10-12 22:54:36.839254 -- Sending packet for message 27900, sequence 0
2019-10-12 22:54:39.072495 -- Sending packet for message 27900, sequence 76
2019-10-12 22:54:41.311265 -- Sending packet for message 27900, sequence 114
```

Kuvio 40. Poikkeusskenaariorissa lähetetyt paketit

Solmulaitteen B vastaanotettua lähetetyt paketit, solmulaite havaitsi lähetyksestä puuttuvan paketin ja pyysi sitä lähetyskantaman sisäpuolella olevilta solmulaitteilta (ks. kuvio 41).

```
2019-10-12 22:54:43.592565 -- Requesting missing packet for message 27900, sequence 38
```

Kuvio 41. Poikkeusskenaariossa puuttuvan paketin pyytäminen

Solmulaite A vastaanotti solmulaitteen B lähettämän pakettipyynnön. Ensin se lokitettiin osaksi solmulaitteen aiemmin lähettämää viestiä, sekvenssinä 38, mutta jatkotarkastuksella paketti huomattiin request-liputetuksi ja siihen osattiin vastata oikealla paketilla (ks. kuvio 42).

```
2019-10-12 22:54:46.377898 -- Received packet for message 27900, sequence 38
2019-10-12 22:54:46.379892 -- messaging::ActionsE::RESPOND
2019-10-12 22:54:46.384905 -- Received packet request for message 27900, sequence 38
2019-10-12 22:54:46.388899 -- Sending packet for message 27900, sequence 38
```

Kuvio 42. Poikkeusskenaariossa pakettipyynnön vastaanotto ja vastaus

Solmulaitteen B vastaanotettua puuttuva paketti, solmulaitteen A lähettämä viesti rakennettiin onnistuneesti (ks. kuvio 43).

```
2019-10-12 22:54:48.645340 -- LoRaModule::onPacket
2019-10-12 22:54:48.649344 -- Received packet for message 27900, sequence 38
2019-10-12 22:54:48.652347 -- messaging::ActionsE::RECEIVE
2019-10-12 22:54:48.657351 -- DownlinkBuffer::StorePacket -- Existing DownlinkPacket found.
2019-10-12 22:54:48.663357 -- DownlinkBuffer::StorePacket -- DownlinkPacket ready to MessageStore.
2019-10-12 22:54:48.666359 -- DownlinkBuffer::SendMessageToStore
2019-10-12 22:54:48.669362 -- SaniaProcessor::HandleMessage
2019-10-12 22:54:48.680372 -- 4236: This message contains 119 characters so it cannot be sent in a single transmission, but must be split across 4 packets.
```

Kuvio 43. Poikkeusskenaariossa puuttuneen paketin vastaanotto ja esitys

Testiskenaarion kulku vastasi skenaarion A määrittelemiä esiehtoja ja toteutti vaaditun lopputilan (ks. luku 2.3.1). Poikkeusskenaariion täydellinen kulku on nähtävissä liitteissä 8 ja 9.

8 Pohdinta

8.1 Prototyypilaitteet

Opinnäytetyön soveltuvuustutkimuksessa toteutettiin prototyypilaitteita, joilla protokollan toimivuutta kyettiin testaamaan käytännössä. Prototyypilaitteiden tarkoitus oli toimia itsenäisinä solmulaitteina verkossa. Prototyypilaitteiden toteutuksessa onnistuttiin siten, että niitä kyettiin lisäämään verkkoon modulaarisesti. Kun opinnäytetyöprotokollan ja prototyypilaitteen koodipohja oli valmis, uuden laitteen käyttöönotto vaati ainoastaan koodin koonnin (engl. compile) prototyypilaitteeseen laitteen sarjaväylää pitkin ja laitteen käynnistyksen koonnin jälkeen. Prototyypilaitteiden toteutuksella voitiin parantaa kehittämistutkimuksen luotettavuutta tuoden lisäarvoa koko opinnäytetyölle.

8.2 Vaatimusten toteutuminen

8.2.1 Toiminnallisuus

Protokolla toteutti kaikki tutkimuksen asettamat vaatimukset kiitettävästi: Data laitteiden välillä saatiin kulkemaan luotettavasti ja virhetilanteista vakaasti palautuen ilman, että laiteryhvät vaikuttivat toistensa toimintaan. Paketit löysivät myös tiensä perille, vaikka yhteisen laiteryhmän solmulaitteet eivät olleet suorassa yhteydessä toisiinsa, vaan paketit välittivät protokollan mukainen mutta samaan laiteryhmään kuulumaton solmulaite.

8.2.2 Tietoturva

Tutkimuksessa keskityttiin pääasiallisesti toimivan tiedonsiirtoprotokollan toteuttamiseen ja siltä osin työn tavoitteet saavutettiin. Koska LoRa-viestit kulkevat kaikkien saavutettavissa olevassa radioverkossa ja viestejä pääsevät lukemaan kaikki ulkopuoliset LoRa-laitteet, jotka konfiguroidaan samoin kuin työn pohjalta luodun verkon solmulaitteet. Näin ollen tosielämän käyttökohteet huomioon ottaen protokollaan täytyy jatkokehityskohteenä toteuttaa laiteryhmäkohtainen

hyötykuorman salausmekanismi, jotta verkkoon kiinnittyneet ulkopuoliset laitteet eivät pääse verkossa liikkuvaan dataan käsiksi.

8.2.3 Suorituskyky

Tiedonsiirto LoRa-tekniikalla on hidasta, mutta vähävirtaista ja lähes virhevapaata. Kehitystutkimuksessa valitulla SF-arvolla 11 yhden toteutetun protokollan mukaisen paketin lähettäminen vei noin 0,76 sekuntia, joka tarkoittaa enimmillään 47 paketin eli 2397 tavun lähettämistä tunneittain, jotta Liikenne- ja viestintäviraston määräämässä rajoituksessa toimintasuhteesta pysytään. Kun paketti lisätään LoRa:n käyttämään pakettikehykseen, lähetysaika kasvaa vielä entisestään tiputtaen efektiivistä pakettien lähetysnopeutta.

8.2.4 Skaalautuvuus

LoRa-tekniikasta ja Liikenne- ja viestintäviraston määräyksistä johtuva protokollan vaatimaton lähetysnopeus tukeekin suuremmalla kantamalla vain laitemäärältään pieneksi rajattuja verkkoja, sillä jokaisen uuden solmulaitteen lisääminen verkkoon laskee yksittäisen laitteen kykyä lähettää uutta dataa verkkoon lineaarisesti tukkien suuremman verkon hyvinkin nopeasti, aiheuttaen pakettien katoamisia. Verkon tukkeutumisen ehkäisemiseksi SF-arvoa on mahdollista laskea, jolloin lähetysnopeus nousee eksponentiaalisesti, joskin tämä tapahtuu lähetyskantaman kustannuksella. Lähettimien konfiguraatiot tulisikin optimoida tarkkaan käyttötapauskohtaisesti, jotta ns. hukkakantamalta vältyttäisiin tiedonsiirron nopeuden maksimoimiseksi.

Vaikka lain määräämien rajoitteiden takia laajan kantaman verkot ovat hitaita ja alttiita tukkeutumisille solmulaitteiden määrän kasvaessa liikaa, on protokolla omiaan toimimaan LoRa-tekniikan tukemilla pienemmillä SF-arvoilla hyvinkin nopeana ja monen yhdistetyn laitteen verkkona pienemmässä skaalassa, kuten kauppakeskusten sensoriverkoissa tai erilaisissa kaupunkien keskusta-alueiden tiedonsiirtojärjestelmissä.

8.3 Protokollan ongelmat ja jatkokehitys

Soveltuvuustutkimus paljasti kulmatapauksen, jossa protokolla palautui virhetilanteesta heikosti silloin, kun END-liputettu paketti jäi saapumatta vastaanottavalle solmulaitteelle. Tällaisia virhetilanteita varten protokollaan tulisi implementoida pakettiajastin, joka havaitsee katkenneen lähetyksen. TCP-protokolla sisältää ajastimen lähetettävälle paketeille, jossa protokollaa käyttävät laitteet lähettävät uuden paketin edellisen tilalle, jos vastaanottava laite ei ole kuitannut lähetettyä pakettia vastaanotetuksi. Opinnäytetyön protokollassa vältettiin protokollan yhteydettömän luonteen vuoksi kuittausviestien käyttöä, mutta vastaava toiminnallisuus kyettäisiin toteuttamaan lisäämällä TCP:ta vastaava ajastin lähetyksen vastaanottavaan päähän. Ajastin voisi toimia siten, että paketit saapuessaan nollaisivat vastaanottavan solmulaitteen ajastimen ja ajastimen saavuttaessa jonkin määritellyn raja-arvon, vastaanottava solmulaite osaisi pyytää uutta pakettia puuttuvan tilalle saapuneiden pakettien sekvenssitietojen perusteella. Ajastimen toteutus on hyvä jatkokehityskohde protokollaan, vaikka se aiheuttaa myös uusia haasteita, kuten kuinka selvittää puuttuuko lähetyksestä viimeistä pakettia ennen muitakin paketteja.

Protokollan monikäyttöisyyttä parantaakseen jatkokehityskohteeksi nostettiin ajatus mahdollisuudesta pyytää korvaava paketti puuttuvan tai viallisen paketin tilalle muiltakin, kuin vain pyytävän solmulaitteen välittömässä läheisyydessä olevilta laitteilta. Paketin pyytäminen ja vastaanotto useampien loikkien takaa vaatisi todennäköisesti TCP-protokollasta tutun TTL-arvon implementoimista tutkimuksen protokollaan, joka kyettäisiin toteuttamaan yhden tavun mittaisella kentällä paketin ylätunnisteessa. Kentän implementointi toisi protokollaan omat loogiset haasteensa, koska paketit eivät verkon mesh-topologisuuden takia liiku verkossa lineaarisesti solmulaitteelta toiselle, vaan voivat saapua sattumanvaraisilta laitteilta. TTL-arvon käyttöönotto mahdollistaisi myös IETF:n kaikuprotokollan (RFC 862:1983) mukaisen toteutuksen, jolloin työn protokolla mahdollistaisi lähetettävien viestien merkitsemisen kaikuviesteiksi ja viestin vastaanottaneet solmulaitteet osaisivat jatkaa kaikuviestin välitystä, vaikka vastaava viesti löytyisikin jo laitteen muistista. Kaikuprotokollalla pystyttäisiin todentamaan protokollan toimivuus ja viestien

muuttumattomuus siirron aikana ja kellottamaan siihen kulunut aika yhden solmulaitteen avulla erilaisissa laitekoonpanoissa.

8.4 Yhteenveto

Opinnäytetyön tavoitteena oli yhdistää radioteknologioita tietoverkko- ja tietoliikenneteknologioihin tutkimalla ja toteuttamalla mesh-verkkotopologinen protokollapino LoRa-verkon päälle kehittämis- ja soveltuvuustutkimuksen avulla. Teoriatasolla jo aikaisessa vaiheessa kävi selväksi, etteivät tämänhetkiset ja laajassa käytössä olevat verkko- tai tiedonsiirtoprotokollat tukeneet vaadittuja toiminnallisuuksia implementoimatta niitä OSI-mallissa sovelluskerrokselle. Näin ollen protokollan levittäminen uusiin laitteisiin olisi hankaloitunut ohjelmistokehittäjän näkökulmasta, vaikuttaen protokollan laajentumiseen negatiivisesti. Siksi muutamasta tämänhetkisestä protokollasta johdettiin teoria uudesta monivaiheista, mesh-verkkotopologista arkkitehtuuria tukevasta verkko- ja tietoliikenneprotokollapinosta (engl. stack) LoRa-verkkoon, joka kykeni myös laajentamaan verkon kantamaa tukemalla solmulaitteiden toimimista rinnan omina laitoryhminään. Kehittämistutkimuksen teorian tukemiseksi toteutetusta pinosta johdettiin soveltuvuustutkimus ja prototyyppilaitte, joilla voitiin vahvistaa niin teorian paikkansapitävyys kuin sen vahvuudet ja heikkoudetkin.

Soveltuvuustutkimuksen tulosten perusteella nähtiin, että LoRa-tekniikalla on mahdollista toteuttaa laaja-alainen tiedonsiirtoverkko. Verkon käytössä on kuitenkin rajoitteensa, kuten tiedonsiirron hidastuminen vaaditun lähetyskantaman kasvaessa sekä Liikenne- ja Viestintäviraston asettamat määräykset verkon solmulaitteiden toimintasuhde- ja säteilytehorajoista. Määräyksiä keventämällä opinnäytetyössä toteutetulla protokollalla kyettäisiin tukemaan suuremman laitemäärän ja useita kilometrejä pitkän kantaman verkkoja, mutta parhaiten se toimii esimerkiksi suurten kauppakeskuksen sensoriverkossa.

Prototyyppilaitteiden lähdekoodi julkaistiin verkossa GitHub-palvelussa MIT-lisenssillä. Opinnäytetyön perusteella LoRa-tekniikan käytöstä ja kehityksestä kiinnostuneet tahot kykenevät arvioimaan ja parantamaan omien ratkaisujensa

toteutusta ja annetun lisenssin puitteissa käyttämään työssä toteutettua protokollaa valmiina tiedonsiirtoratkaisuna verkoissaan. Johtaen LoRa-tekniikan kehitystä eteenpäin, työ täytti tavoitteensa.

Lähteet

15 AO/2019 M. Määräys luvasta vapaiden radiolähettimien yhteistaajuuksista ja käytöstä. Annettu 09.01.2019. Viitattu 05.10.2019.

<https://www.finlex.fi/fi/viranomaiset/normi/480001/44836>

Bahl, M. 2018. OSI Model Layers — “Explained”. Artikkele Medium-verkkosivulla. Viitattu 01.09.2019. <https://medium.com/learn-with-the-lean-programmer/osi-model-layers-explained-ee1d43058c1f>

Chauhan, R. & Lee, K. 2018. 11 Myths About LoRaWAN. Artikkele ElectronicDesign - verkkosivulla. Viitattu 09.10.2019. <https://www.electronicdesign.com/industrial-automation/11-myths-about-lorawan>

Data Rate and Spreading Factor. N.d. Artikkele Exploratory Engineering -verkkosivulla. Viitattu 12.10.2019. https://docs.exploratory.engineering/lora/dr_sf/

Effective Isotropic Radiated Power (EIRP). N.d. Artikkele antenna-theory - verkkosivulla. Viitattu 08.11.2019. <http://www.antenna-theory.com/definitions/eirp.php>

ESP32 Series Datasheet. 2019. Espressif Systemsin tekninen esite. Viitattu 05.10.2019.

https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

FIFO (First-In-First-Out) approach in Programming. N.d. Artikkele GeeksForGeeks - verkkosivulla. Viitattu 23.6.2019. <https://www.geeksforgeeks.org/fifo-first-in-first-out-approach-in-programming/>

Fox, P. N.d. Transmission Control Protocol (TCP). Artikkele Khan Academy - verkkosivulla. Viitattu 14.08.2019. <https://www.khanacademy.org/computing/ap-computer-science-principles/the-internet/tcp-fault-tolerant-transmission-protocol/a/transmission-control-protocol-tcp>

Gregg, M. 2006. Hack the Stack. Syngress Publishing Inc.

Grilo, A. 2018. LoRaWAN: An Introduction. PowerPoint -presentaatio Lissabonin yliopiston verkkosivulla. Viitattu 12.10.2019.

<https://fenix.tecnico.ulisboa.pt/downloadFile/1970943312328262/LoRaWAN%20Introduction.pdf>

Guidelines for determining the effective radiated power (ERP) and equivalent isotropically radiated power (EIRP) of an RF transmitting system. 2015. Federal Communications Commission antama ohjeistus. Viitattu 03.11.2019.

<https://apps.fcc.gov/kdb/GetAttachment.html?id=fzlsGm%2Fe68Ymx58IAmzNbw%3D%3D&desc=412172%20D01%20Determining%20ERP%20and%20EIRP%20v01r01>

Half Wave Dipole Antenna / Aerial. N.n. Artikkelit Electronics Notes -verkkosivulla. Viitattu 02.11.2019. <https://www.electronics-notes.com/articles/antennas-propagation/dipole-antenna/half-wave-dipole.php>

Hamrioui, S., Mona-Cruz, E., Mroue, H., Nasser, A., Parrein, B. & Rouyer, G. 2018. Analytical and Simulation study for LoRa Modulation. 2018 25th International Conference on Telecommunications (ICT). IEEE Xplore Digital Library. Viitattu 14.10.2019. <https://ieeexplore.ieee.org/document/8464879/>

Hanson, R. 2011. Common Pitfalls – Don't Be A Victim. Artikkelit GitHub -verkkosivulla. Viitattu 19.08.2019. <https://github.com/robbiehanson/CocoaAsyncSocket/wiki/CommonPitfalls>

How Flash Memory Works – Advantages & Disadvantages. 2018. Artikkelit Arrow -verkkosivulla. Viitattu 05.10.2019. <https://www.arrow.com/en/research-and-events/articles/how-flash-memory-works-advantages-and-disadvantages>

Hyöky, A. & Kyllönen, E. 2013. Lukuhöperöksi kasvamassa – Lukupassimateriaalin kehittäminen esiopetukseen. Kasvatustieteen pro gradu -tutkielma. Viitattu 08.11.2019. https://www.ouka.fi/c/document_library/get_file?uuid=23e10170-d394-414d-8102-791bf4ac3fd4&groupId=112792

Ianelli, Z. 2003. Introduction to Chirp Spread Spectrum (CSS) Technology. PowerPoint -presentaatio IEEE 8020 LAN/MAN standardikomitean verkkosivustolla. Viitattu 22.07.2019. http://www.ieee802.org/802_tutorials/03-November/15-03-0460-00-0040-IEEE-802-CSS-Tutorial-part1.ppt

IETF RFC 862. 1983. Internet Engineering Task Forcen määrittämä standardi. Viitattu 30.09.2019. <https://tools.ietf.org/html/rfc862>

Introduction to LoRa Technology – The Game Changer. 2016. Artikkelit Maker.io -verkkosivulla. Viitattu 03.11.2019. <https://www.digikey.com/en/maker/blogs/introduction-to-lora-technology>

Introduction to LoRa. N.d. Artikkelit Multitech -verkkosivulla. Viitattu 18.08.2019. <http://www.multitech.net/developer/software/lora/introduction-to-lora/>

Iveson, S. 2019. IP Time to Live (TTL) and Hop Limit Basics. Artikkelit Packet Pushers -verkkosivulla. Viitattu 02.11.2019. <https://packetpushers.net/ip-time-to-live-and-hop-limit-basics/>

Knight, M. N.d. REVERSING LORA. Presentaatio Squarespace-verkkosivulla. Viitattu 07.10.2019. <https://static1.squarespace.com/static/54cece7e4b054df1848b5f9/t/57489e6e07eaa0105215dc6c/1464376943218/Reversing-Lora-Knight.pdf>

Mistry, S. N.d. LoRa API. arduino-LoRa -kirjaston rajapintadokumentaatio GitHub -repositoriossa. Muokattu 19.08.2018. Viitattu 13.10.2019. <https://github.com/sandeepmistry/arduino-LoRa/blob/master/API.md>

- Mitchell, B. 2019. User Datagram Protocol – Understanding UDP and How It’s Different From TCP. Artikkele Lifewire -verkkosivulla. Viitattu 14.08.2019. <https://www.lifewire.com/user-datagram-protocol-817976>
- Mitchell, B. N.d. What Are Network Protocols?. Artikkele LifeWire -verkkosivulla. Päivitetty 28.09.2019. Viitattu 26.10.2019. <https://www.lifewire.com/definition-of-protocol-network-817949>
- Neagu, C. 2019. Simple questions: What is a SoC (System on a Chip)? Artikkele Digital Citizen -verkkosivulla. Viitattu 12.10.2019. <https://www.digitalcitizen.life/soc-system-on-chip>
- Nyberg, J. & Jokela, K. 2006. Sähkömagneettiset kentät. Helsinki: Säteilyturvakeskus. <http://urn.fi/URN:NBN:fi-fe2014120247384>
- Pal, S. N.d. Different Types of RAM (Random Access Memory). Artikkele GeeksForGeeks -verkkosivulla. Viitattu 05.10.2019. <https://www.geeksforgeeks.org/different-types-ram-random-access-memory/>
- PlatformIO is an open source ecosystem for IoT development. N.d. Dokumentaatio PlatformIO-ympäristöstä platformio.org-verkkosivulla. Viitattu 07.10.2019. <https://docs.platformio.org/en/latest/what-is-platformio.html>
- Pollin, S & Reynders, B. 2016. Chirp spread spectrum as a modulation technique for long range communication. 2016 Symposium on Communications and Vehicular Technologies (SCVT). IEEE Xplore Digital Library. Viitattu 01.09.2019. <https://ieeexplore.ieee.org/document/7797659/>
- Radioaallot ympäristössämme. 2009. Säteilyturvakeskuksen julkaisema artikkele. Viitattu 07.11.2019. <http://urn.fi/URN:NBN:fi-fe2014120249574>
- Radiolaitteiden vaatimustenmukaisuus takaa toimivan radioliikenteen. 2019. Artikkele Liikenne- ja viestintäviraston verkkosivulla. Viitattu 07.11.2019. <https://www.traficom.fi/fi/viestinta/viestintaverkot/radiolaitteiden-vaatimustenmukaisuus-takaa-toimivan-radioliikenteen>
- Reference Models in Communication Networks. N.d. Artikkele Studytonight -verkkosivulla. Viitattu 07.11.2019. <https://www.studytonight.com/computer-networks/reference-models>
- RFC4122. 2005. A Universally Unique Identifier (UUID) URN Namespace. Viitattu 22.07.2019. <https://www.ietf.org/rfc/rfc4122.txt>
- Rouse, M. & Shea, S. N.d. Microcontroller. Artikkele TechTarget -verkkosivulla. Viitattu 06.11.2019. <https://internetofthingsagenda.techtarget.com/definition/microcontroller>
- Rouse, M. N.d. Radio frequency (RF, rf). Artikkele TechTarget -verkkosivulla. Viitattu 08.10.2019. <https://searchnetworking.techtarget.com/definition/radio-frequency>
- Shekhar, A. 2016. What Is Mesh Topology? Advantages And Disadvantages Of Mesh Topology. Artikkele Fossbytes -verkkosivulla. Viitattu 9.3.2019.

<https://fossbytes.com/what-is-mesh-topology-advantages-and-disadvantages-of-mesh-topology/>

Silver, H. 2018. Effective Radiated Power. Julkaisu Ham Radio Science Citizen Investigation -verkkosivulla. Viitattu 08.10.2019.
<https://hamsci.org/sites/default/files/resources/erp.pdf>

Stallings, W. 2013. Data and Computer Communications. Tenth Edition. New Jersey: Pearson Education, inc.

SX1276/77/78/79 - 137 MHz to 1020 MHz Low Power Long Range Transceiver. 2019. Tekninen esite. Viitattu 05.10.2019.
<https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R0000001OKx/JUYM3TvBMenQzU4LS8ZJcM58BljCcoZcUpHV0gnZ.y0>

Tewari, D. N.d. Introduction of Microprocessor. Artikkelit GeeksForGeeks - verkkosivulla. Viitattu 12.09.2019. <https://www.geeksforgeeks.org/introduction-of-microprocessor/>

THIRU (nimimerkki). N.d. Error Control in TCP. Oppimateriaali My Reading Room - verkkosivulla. Viitattu 12.10.2019. <http://www.myreadingroom.co.in/notes-and-studymaterial/68-dcn/853-error-control-in-tcp.html>

Tyson, J. N.d. How Flash Memory Works. Artikkelit HowStuffWorks -verkkosivulla. Viitattu 26.06.2019. <https://computer.howstuffworks.com/flash-memory.htm>

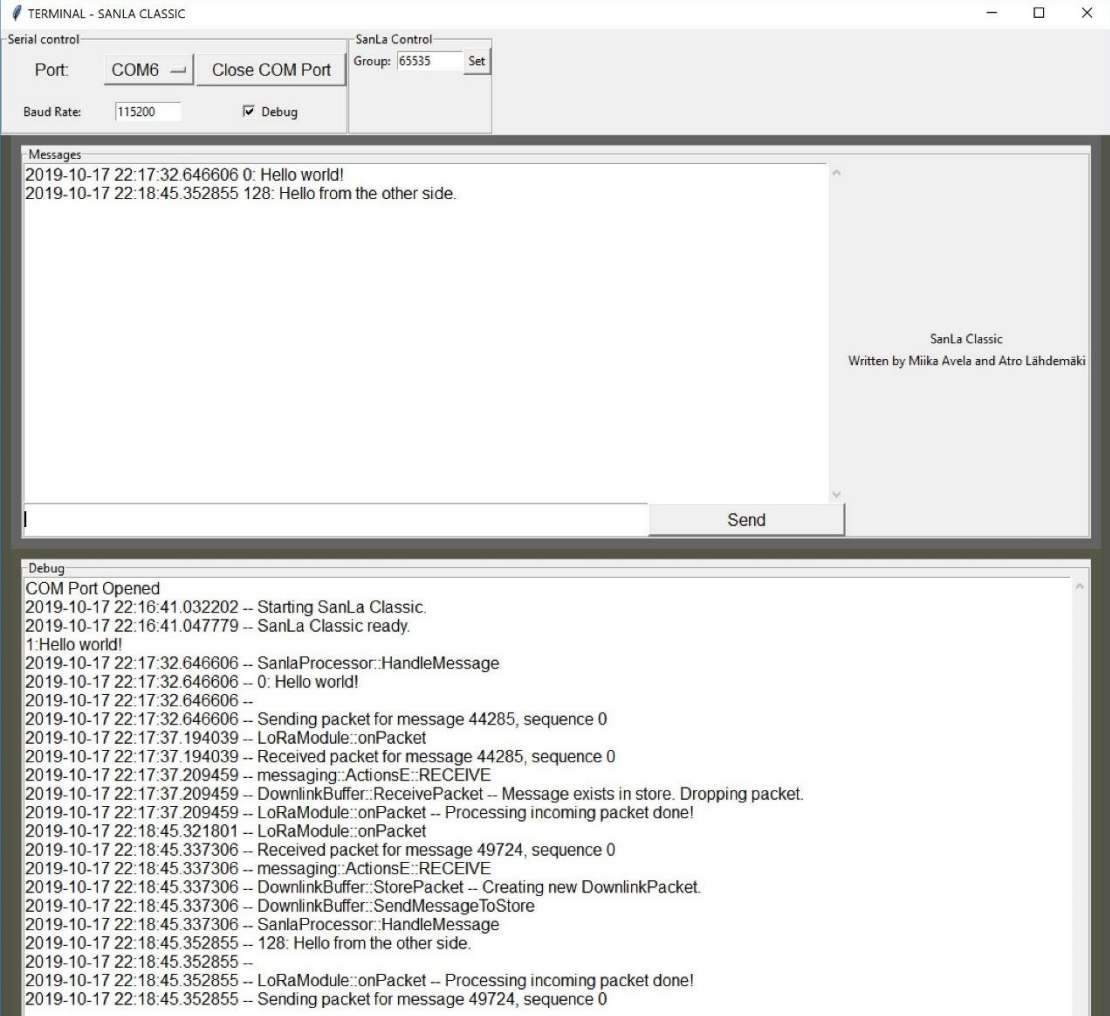
Unuth, N. 2018. TCP (Transmission Control Protocol) Explained. Lifewire. Viitattu 14.08.2019. <https://www.lifewire.com/tcp-transmission-control-protocol-3426736>

Weil, N. 2012. Q&A: What is a development board? Artikkelit nickweil-verkkosivulla. Viitattu 31.08.2019. <http://www.nickweil.com/2012/01/q-what-is-development-board.html>

What is LoRa®? N.d. Artikkelit Semtech:n verkkosivulla. Viitattu 08.11.2019.
<https://www.semtech.com/lora/what-is-lora>

Liitteet

Liite 1. Yhteystesti: Laitteen A loki



```
TERMINAL - SANLA CLASSIC

Serial control
Port: COM6 Close COM Port
Baud Rate: 115200 Debug
SanLa Control
Group: 65535 Set

Messages
2019-10-17 22:17:32.646606 0: Hello world!
2019-10-17 22:18:45.352855 128: Hello from the other side.

SanLa Classic
Written by Miika Avela and Atro Lähdemäki

Send

Debug
COM Port Opened
2019-10-17 22:16:41.032202 -- Starting SanLa Classic.
2019-10-17 22:16:41.047779 -- SanLa Classic ready.
1:Hello world!
2019-10-17 22:17:32.646606 -- SanLaProcessor::HandleMessage
2019-10-17 22:17:32.646606 -- 0: Hello world!
2019-10-17 22:17:32.646606 --
2019-10-17 22:17:32.646606 -- Sending packet for message 44285, sequence 0
2019-10-17 22:17:37.194039 -- LoRaModule::onPacket
2019-10-17 22:17:37.194039 -- Received packet for message 44285, sequence 0
2019-10-17 22:17:37.209459 -- messaging::ActionsE::RECEIVE
2019-10-17 22:17:37.209459 -- DownlinkBuffer::ReceivePacket -- Message exists in store. Dropping packet.
2019-10-17 22:17:37.209459 -- LoRaModule::onPacket -- Processing incoming packet done!
2019-10-17 22:18:45.321801 -- LoRaModule::onPacket
2019-10-17 22:18:45.337306 -- Received packet for message 49724, sequence 0
2019-10-17 22:18:45.337306 -- messaging::ActionsE::RECEIVE
2019-10-17 22:18:45.337306 -- DownlinkBuffer::StorePacket -- Creating new DownlinkPacket.
2019-10-17 22:18:45.337306 -- DownlinkBuffer::SendMessageToStore
2019-10-17 22:18:45.337306 -- SanLaProcessor::HandleMessage
2019-10-17 22:18:45.352855 -- 128: Hello from the other side.
2019-10-17 22:18:45.352855 --
2019-10-17 22:18:45.352855 -- LoRaModule::onPacket -- Processing incoming packet done!
2019-10-17 22:18:45.352855 -- Sending packet for message 49724, sequence 0
```

Liite 2. Yhteystesti: Laitteen B loki

The screenshot shows the SanLa Classic terminal window. The top bar is titled "TERMINAL - SANLA CLASSIC". Below it, there are controls for "Serial control" (Port: /dev/ttyUSB0, Baud Rate: 115200, Debug checked) and "SanLa Control" (Group: 65535, Set button). The main area is split into two panes: "Messages" and "Debug".

Messages pane:

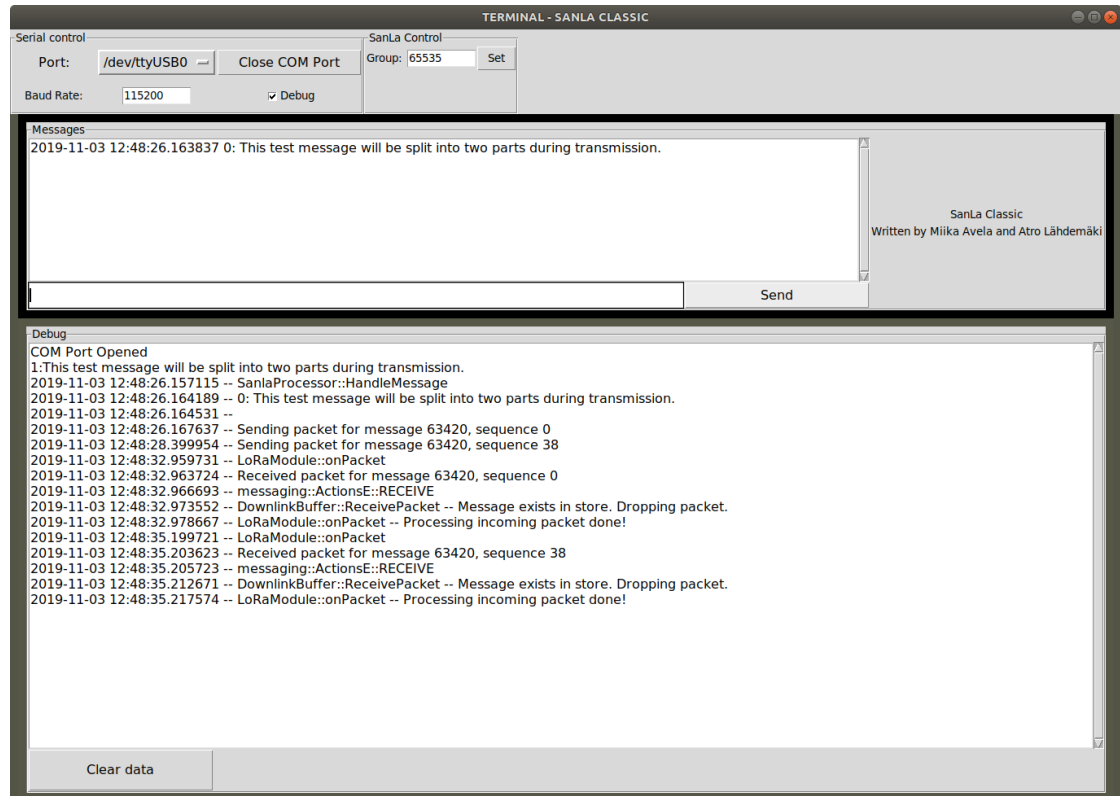
```
2019-10-17 22:17:34.515036 62612: Hello world!  
2019-10-17 22:18:42.643831 0: Hello from the other side.
```

Debug pane:

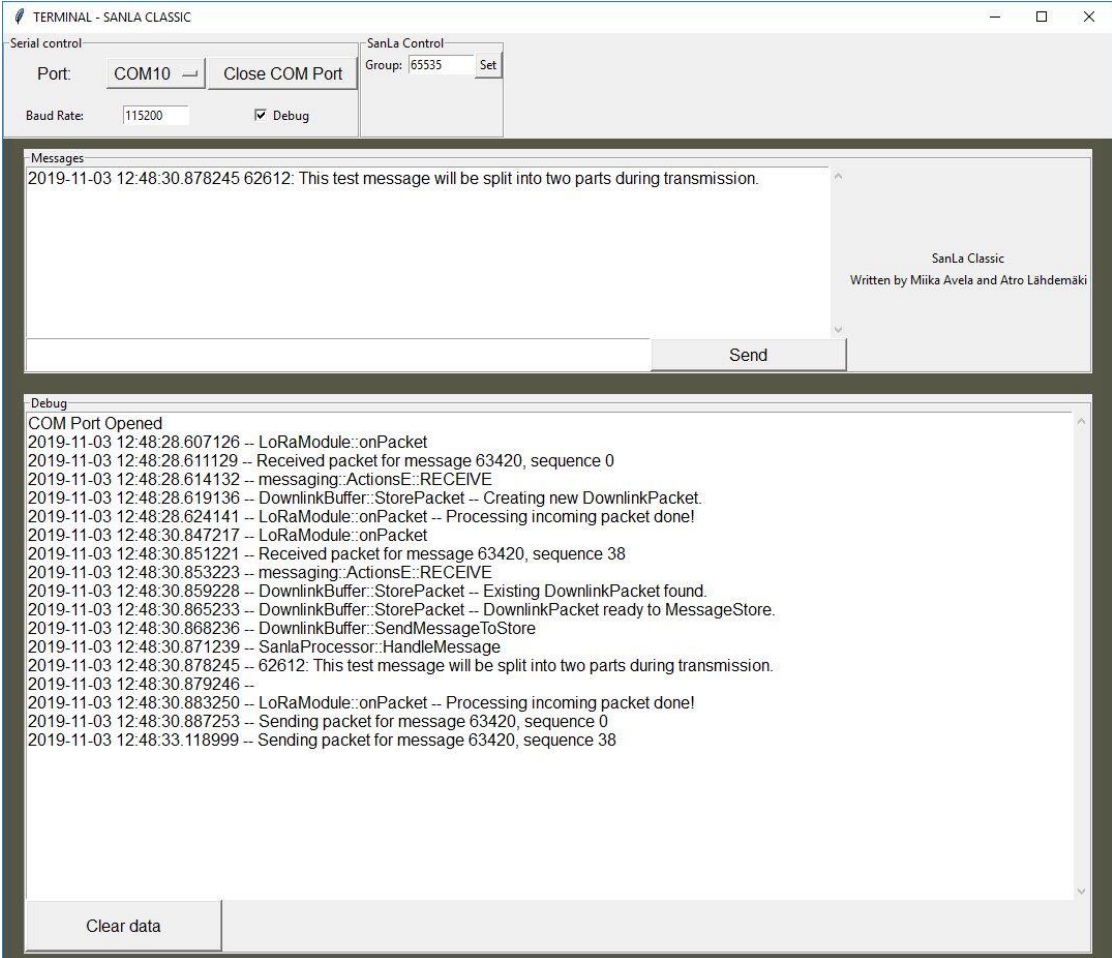
```
COM Port Opened  
2019-10-17 22:17:34.495314 -- LoRaModule::onPacket  
2019-10-17 22:17:34.499967 -- Received packet for message 44285, sequence 0  
2019-10-17 22:17:34.502025 -- messaging::ActionsE::RECEIVE  
2019-10-17 22:17:34.507056 -- DownlinkBuffer::StorePacket -- Creating new DownlinkPacket.  
2019-10-17 22:17:34.511071 -- DownlinkBuffer::SendMessageToStore  
2019-10-17 22:17:34.513115 -- SanlaProcessor::HandleMessage  
2019-10-17 22:17:34.515273 -- 62612: Hello world!  
2019-10-17 22:17:34.515612 --  
2019-10-17 22:17:34.519977 -- LoRaModule::onPacket -- Processing incoming packet done!  
2019-10-17 22:17:34.523992 -- Sending packet for message 44285, sequence 0  
1:Hello from the other side.  
2019-10-17 22:18:42.641684 -- SanlaProcessor::HandleMessage  
2019-10-17 22:18:42.644196 -- 0: Hello from the other side.  
2019-10-17 22:18:42.644587 --  
2019-10-17 22:18:42.648538 -- Sending packet for message 49724, sequence 0  
2019-10-17 22:18:47.192667 -- LoRaModule::onPacket  
2019-10-17 22:18:47.196704 -- Received packet for message 49724, sequence 0  
2019-10-17 22:18:47.199669 -- messaging::ActionsE::RECEIVE  
2019-10-17 22:18:47.206627 -- DownlinkBuffer::ReceivePacket -- Message exists in store. Dropping packet.  
2019-10-17 22:18:47.211652 -- LoRaModule::onPacket -- Processing incoming packet done!
```

On the right side of the Messages pane, there is a small text box: "SanLa Classic Written by Miika Avela and Aro Lähdemäki". A "Send" button is located at the bottom right of the Messages pane.

Liite 3. Skenaario A: Laitteen A loki



Liite 4. Skenaario A: Laitteen B loki



TERMINAL - SANLA CLASSIC

Serial control: Port: COM10 Close COM Port Baud Rate: 115200 Debug

SanLa Control: Group: 65535 Set

Messages

2019-11-03 12:48:30.878245 62612: This test message will be split into two parts during transmission.

SanLa Classic
Written by Miika Avela and Atro Lähdemäki

Send

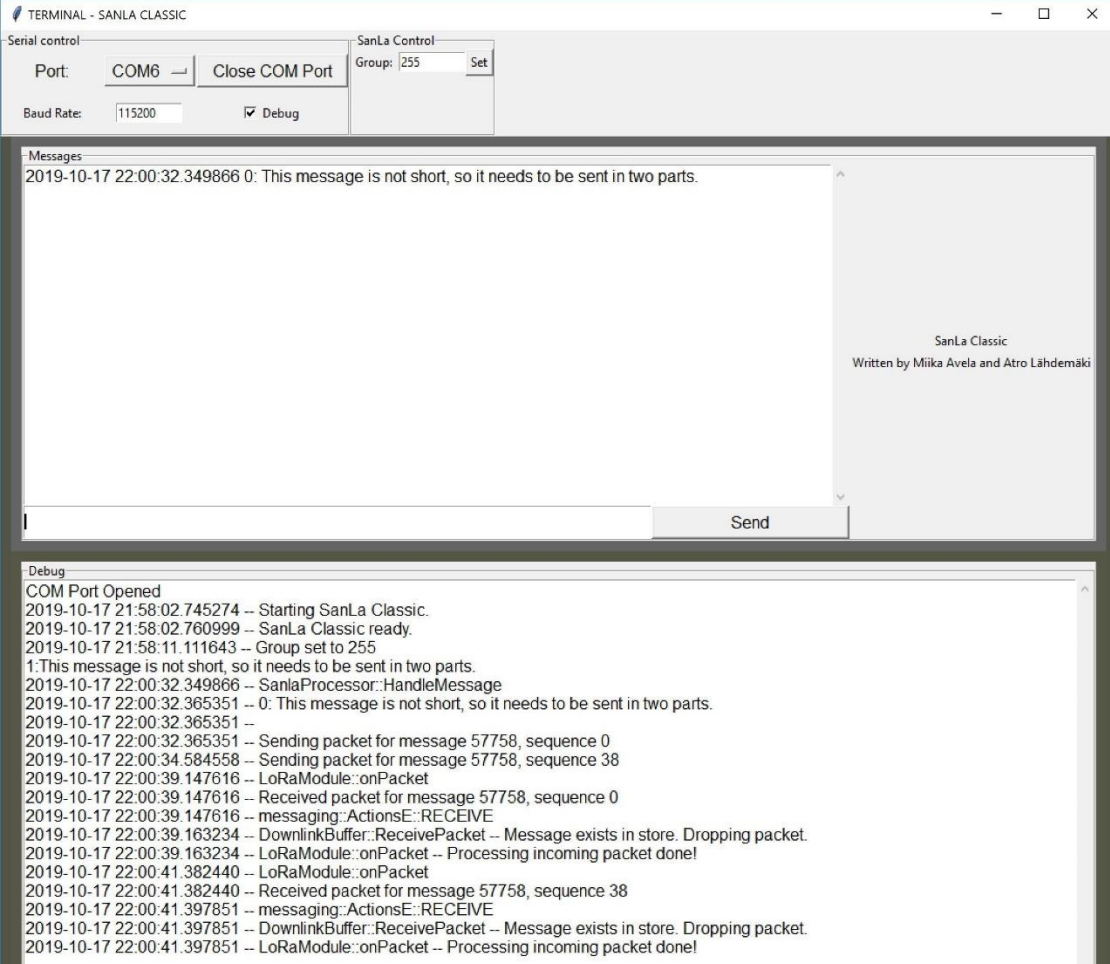
Debug

COM Port Opened

```
2019-11-03 12:48:28.607126 -- LoRaModule::onPacket
2019-11-03 12:48:28.611129 -- Received packet for message 63420, sequence 0
2019-11-03 12:48:28.614132 -- messaging::ActionsE::RECEIVE
2019-11-03 12:48:28.619136 -- DownlinkBuffer::StorePacket -- Creating new DownlinkPacket.
2019-11-03 12:48:28.624141 -- LoRaModule::onPacket -- Processing incoming packet done!
2019-11-03 12:48:30.847217 -- LoRaModule::onPacket
2019-11-03 12:48:30.851221 -- Received packet for message 63420, sequence 38
2019-11-03 12:48:30.853223 -- messaging::ActionsE::RECEIVE
2019-11-03 12:48:30.859228 -- DownlinkBuffer::StorePacket -- Existing DownlinkPacket found.
2019-11-03 12:48:30.865233 -- DownlinkBuffer::StorePacket -- DownlinkPacket ready to MessageStore.
2019-11-03 12:48:30.868236 -- DownlinkBuffer::SendMessageToStore
2019-11-03 12:48:30.871239 -- SanLaProcessor::HandleMessage
2019-11-03 12:48:30.878245 -- 62612: This test message will be split into two parts during transmission.
2019-11-03 12:48:30.879246 --
2019-11-03 12:48:30.883250 -- LoRaModule::onPacket -- Processing incoming packet done!
2019-11-03 12:48:30.887253 -- Sending packet for message 63420, sequence 0
2019-11-03 12:48:33.118999 -- Sending packet for message 63420, sequence 38
```

Clear data

Liite 5. Skenaario B: Laitteen A loki



The screenshot displays the 'TERMINAL - SANLA CLASSIC' window. At the top, there are control panels for 'Serial control' and 'SanLa Control'. The 'Serial control' panel shows 'Port: COM6', 'Close COM Port', 'Baud Rate: 115200', and a checked 'Debug' option. The 'SanLa Control' panel shows 'Group: 255' and a 'Set' button. Below these panels is a 'Messages' section with a text input area containing the message: '2019-10-17 22:00:32.349866 0: This message is not short, so it needs to be sent in two parts.' A 'Send' button is located at the bottom right of this section. To the right of the message input area, there is a vertical sidebar with the text: 'SanLa Classic' and 'Written by Miika Avela and Aatro Lähdemäki'. At the bottom of the terminal window is a 'Debug' section containing a log of system events, including 'COM Port Opened', 'Starting SanLa Classic', 'SanLa Classic ready', 'Group set to 255', and several messages related to packet sending and receiving for message 57758.

Serial control: Port: COM6 Close COM Port Baud Rate: 115200 Debug

SanLa Control: Group: 255 Set

Messages

2019-10-17 22:00:32.349866 0: This message is not short, so it needs to be sent in two parts.

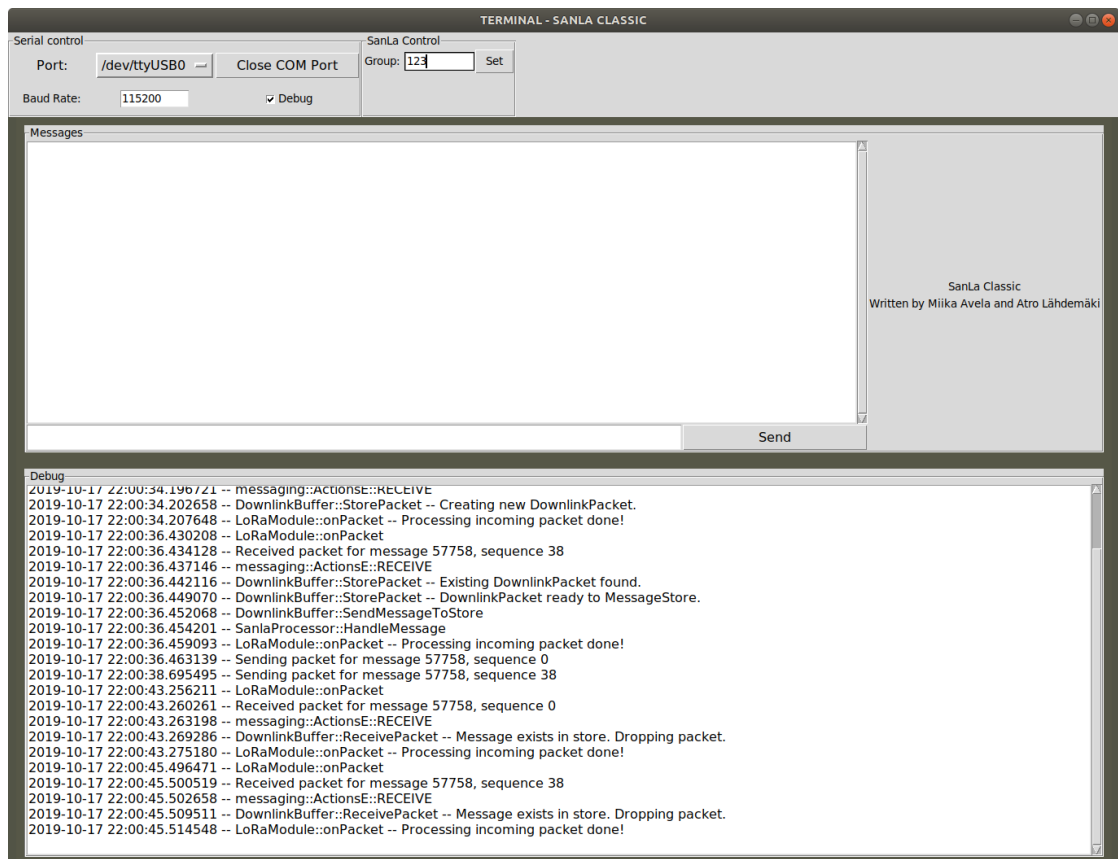
SanLa Classic
Written by Miika Avela and Aatro Lähdemäki

Send

Debug

COM Port Opened
2019-10-17 21:58:02.745274 -- Starting SanLa Classic.
2019-10-17 21:58:02.760999 -- SanLa Classic ready.
2019-10-17 21:58:11.111643 -- Group set to 255
1: This message is not short, so it needs to be sent in two parts.
2019-10-17 22:00:32.349866 -- SanLaProcessor::HandleMessage
2019-10-17 22:00:32.365351 -- 0: This message is not short, so it needs to be sent in two parts.
2019-10-17 22:00:32.365351 --
2019-10-17 22:00:32.365351 -- Sending packet for message 57758, sequence 0
2019-10-17 22:00:34.584558 -- Sending packet for message 57758, sequence 38
2019-10-17 22:00:39.147616 -- LoRaModule::onPacket
2019-10-17 22:00:39.147616 -- Received packet for message 57758, sequence 0
2019-10-17 22:00:39.147616 -- messaging::ActionsE::RECEIVE
2019-10-17 22:00:39.163234 -- DownlinkBuffer::ReceivePacket -- Message exists in store. Dropping packet.
2019-10-17 22:00:39.163234 -- LoRaModule::onPacket -- Processing incoming packet done!
2019-10-17 22:00:41.382440 -- LoRaModule::onPacket
2019-10-17 22:00:41.382440 -- Received packet for message 57758, sequence 38
2019-10-17 22:00:41.397851 -- messaging::ActionsE::RECEIVE
2019-10-17 22:00:41.397851 -- DownlinkBuffer::ReceivePacket -- Message exists in store. Dropping packet.
2019-10-17 22:00:41.397851 -- LoRaModule::onPacket -- Processing incoming packet done!

Liite 6. Skenaario B: Laitteen B loki



The screenshot shows the SanLa Classic terminal interface. At the top, it is titled "TERMINAL - SANLA CLASSIC". Below the title bar, there are two main sections: "Serial control" and "SanLa Control".

Serial control:

- Port: /dev/ttyUSB0
- Close COM Port
- Baud Rate: 115200
- Debug:

SanLa Control:

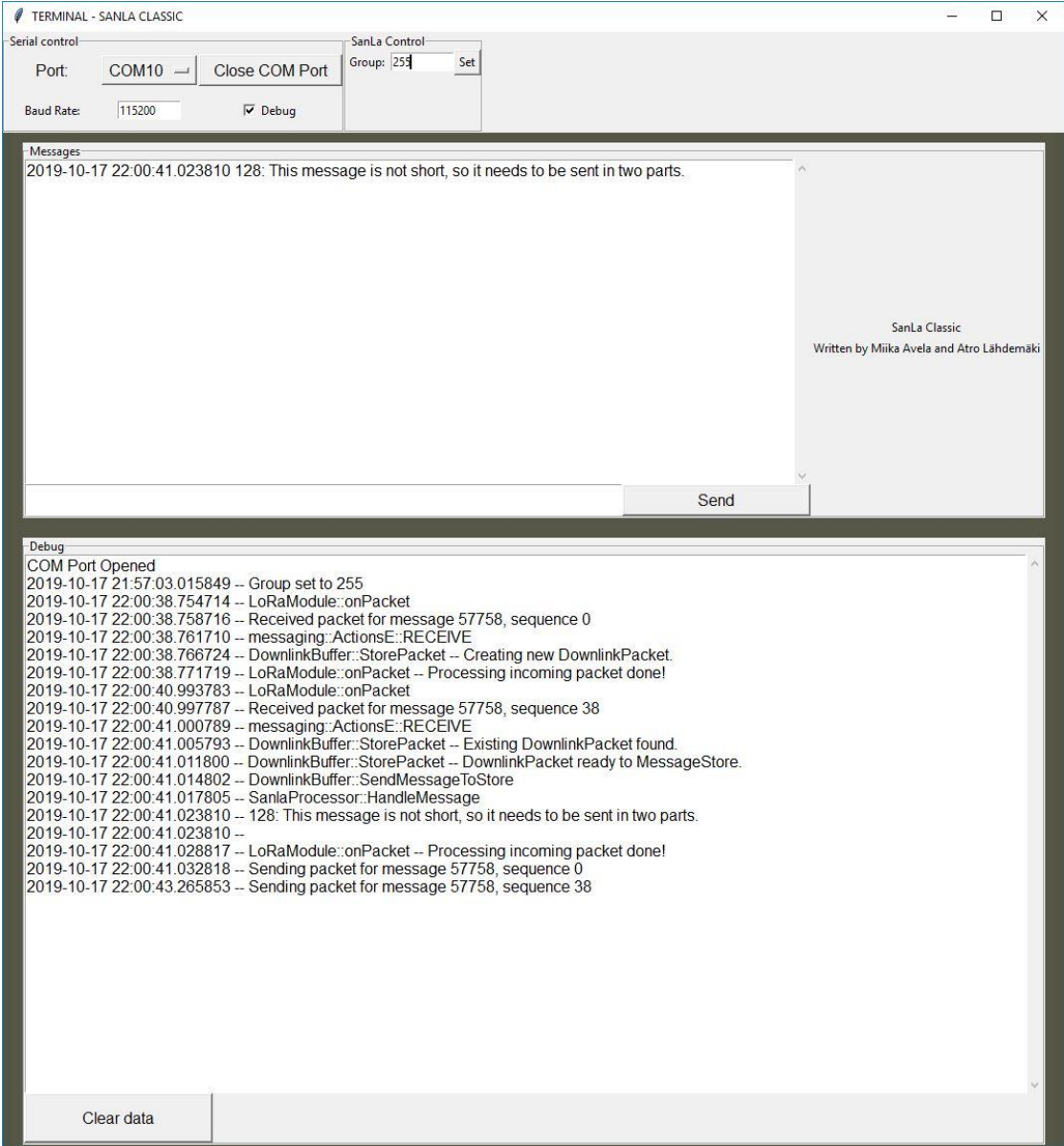
- Group: 123
- Set

The main area is divided into two panes. The top pane is labeled "Messages" and is currently empty. The bottom pane is labeled "Debug" and contains a log of system events:

```
2019-10-17 22:00:34.196721 -- messaging::ActionSE::RECEIVE
2019-10-17 22:00:34.202658 -- DownlinkBuffer::StorePacket -- Creating new DownlinkPacket.
2019-10-17 22:00:34.207648 -- LoRaModule::onPacket -- Processing incoming packet done!
2019-10-17 22:00:36.430208 -- LoRaModule::onPacket
2019-10-17 22:00:36.434128 -- Received packet for message 57758, sequence 38
2019-10-17 22:00:36.437146 -- messaging::ActionSE::RECEIVE
2019-10-17 22:00:36.442116 -- DownlinkBuffer::StorePacket -- Existing DownlinkPacket found.
2019-10-17 22:00:36.449070 -- DownlinkBuffer::StorePacket -- DownlinkPacket ready to MessageStore.
2019-10-17 22:00:36.452068 -- DownlinkBuffer::SendMessageToStore
2019-10-17 22:00:36.454201 -- SanLaProcessor::HandleMessage
2019-10-17 22:00:36.459093 -- LoRaModule::onPacket -- Processing incoming packet done!
2019-10-17 22:00:36.463139 -- Sending packet for message 57758, sequence 0
2019-10-17 22:00:38.695495 -- Sending packet for message 57758, sequence 38
2019-10-17 22:00:43.256211 -- LoRaModule::onPacket
2019-10-17 22:00:43.260261 -- Received packet for message 57758, sequence 0
2019-10-17 22:00:43.263198 -- messaging::ActionSE::RECEIVE
2019-10-17 22:00:43.269286 -- DownlinkBuffer::ReceivePacket -- Message exists in store. Dropping packet.
2019-10-17 22:00:43.275180 -- LoRaModule::onPacket -- Processing incoming packet done!
2019-10-17 22:00:45.496471 -- LoRaModule::onPacket
2019-10-17 22:00:45.500519 -- Received packet for message 57758, sequence 38
2019-10-17 22:00:45.502658 -- messaging::ActionSE::RECEIVE
2019-10-17 22:00:45.509511 -- DownlinkBuffer::ReceivePacket -- Message exists in store. Dropping packet.
2019-10-17 22:00:45.514548 -- LoRaModule::onPacket -- Processing incoming packet done!
```

On the right side of the terminal, there is a small text box that reads: "SanLa Classic Written by Miika Avela and Aro Lähdemäki".

Liite 7. Skenaario B: Laitteen C loki



TERMINAL - SANLA CLASSIC

Serial control: Port: COM10 Close COM Port Baud Rate: 115200 Debug

SanLa Control: Group: 255 Set

Messages

2019-10-17 22:00:41.023810 128: This message is not short, so it needs to be sent in two parts.

SanLa Classic
Written by Miika Avela and Atro Lähdenmäki

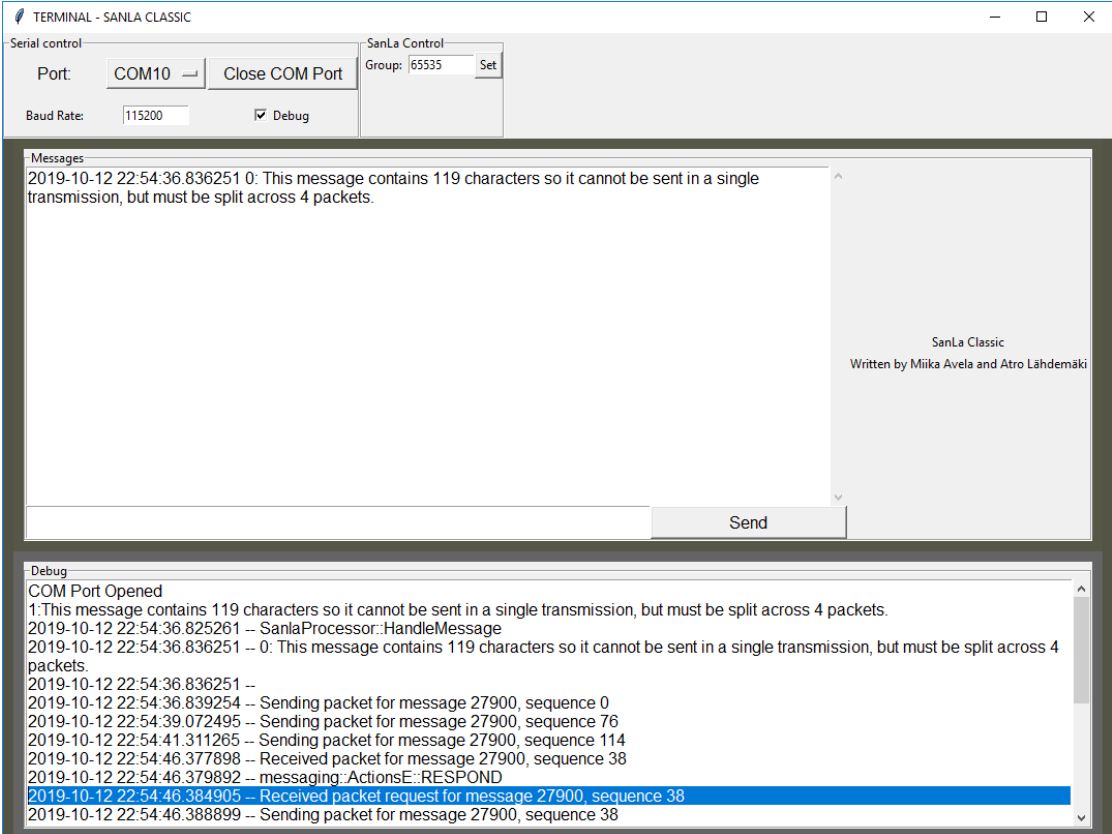
Send

Debug

```
COM Port Opened
2019-10-17 21:57:03.015849 -- Group set to 255
2019-10-17 22:00:38.754714 -- LoRaModule::onPacket
2019-10-17 22:00:38.758716 -- Received packet for message 57758, sequence 0
2019-10-17 22:00:38.761710 -- messaging::ActionsE::RECEIVE
2019-10-17 22:00:38.766724 -- DownlinkBuffer::StorePacket -- Creating new DownlinkPacket
2019-10-17 22:00:38.771719 -- LoRaModule::onPacket -- Processing incoming packet done!
2019-10-17 22:00:40.993783 -- LoRaModule::onPacket
2019-10-17 22:00:40.997787 -- Received packet for message 57758, sequence 38
2019-10-17 22:00:41.000789 -- messaging::ActionsE::RECEIVE
2019-10-17 22:00:41.005793 -- DownlinkBuffer::StorePacket -- Existing DownlinkPacket found.
2019-10-17 22:00:41.011800 -- DownlinkBuffer::StorePacket -- DownlinkPacket ready to MessageStore.
2019-10-17 22:00:41.014802 -- DownlinkBuffer::SendMessageToStore
2019-10-17 22:00:41.017805 -- SanlaProcessor::HandleMessage
2019-10-17 22:00:41.023810 -- 128: This message is not short, so it needs to be sent in two parts.
2019-10-17 22:00:41.023810 --
2019-10-17 22:00:41.028817 -- LoRaModule::onPacket -- Processing incoming packet done!
2019-10-17 22:00:41.032818 -- Sending packet for message 57758, sequence 0
2019-10-17 22:00:43.265853 -- Sending packet for message 57758, sequence 38
```

Clear data

Liite 8. Poikkeuskenaario: Laitteen A loki



The screenshot shows the SanLa Classic terminal interface. At the top, there are controls for the serial port: Port (COM10), Close COM Port, Baud Rate (115200), and Debug (checked). The Messages pane shows a message received at 2019-10-12 22:54:36.836251 0: "This message contains 119 characters so it cannot be sent in a single transmission, but must be split across 4 packets." The Debug pane shows a sequence of events: "COM Port Opened", "1: This message contains 119 characters so it cannot be sent in a single transmission, but must be split across 4 packets.", "2019-10-12 22:54:36.825261 -- SanlaProcessor::HandleMessage", "2019-10-12 22:54:36.836251 -- 0: This message contains 119 characters so it cannot be sent in a single transmission, but must be split across 4 packets.", "2019-10-12 22:54:36.836251 --", "2019-10-12 22:54:36.839254 -- Sending packet for message 27900, sequence 0", "2019-10-12 22:54:39.072495 -- Sending packet for message 27900, sequence 76", "2019-10-12 22:54:41.311265 -- Sending packet for message 27900, sequence 114", "2019-10-12 22:54:46.377898 -- Received packet for message 27900, sequence 38", "2019-10-12 22:54:46.379892 -- messaging::ActionsE::RESPOND", "2019-10-12 22:54:46.384905 -- Received packet request for message 27900, sequence 38", and "2019-10-12 22:54:46.388899 -- Sending packet for message 27900, sequence 38".

TERMINAL - SANLA CLASSIC

Serial control

Port: COM10 Close COM Port

Baud Rate: 115200 Debug

SanLa Control

Group: 65535 Set

Messages

2019-10-12 22:54:36.836251 0: This message contains 119 characters so it cannot be sent in a single transmission, but must be split across 4 packets.

SanLa Classic
Written by Miika Avela and Atro Lähdemäki

Send

Debug

COM Port Opened

1: This message contains 119 characters so it cannot be sent in a single transmission, but must be split across 4 packets.

2019-10-12 22:54:36.825261 -- SanlaProcessor::HandleMessage

2019-10-12 22:54:36.836251 -- 0: This message contains 119 characters so it cannot be sent in a single transmission, but must be split across 4 packets.

2019-10-12 22:54:36.836251 --

2019-10-12 22:54:36.839254 -- Sending packet for message 27900, sequence 0

2019-10-12 22:54:39.072495 -- Sending packet for message 27900, sequence 76

2019-10-12 22:54:41.311265 -- Sending packet for message 27900, sequence 114

2019-10-12 22:54:46.377898 -- Received packet for message 27900, sequence 38

2019-10-12 22:54:46.379892 -- messaging::ActionsE::RESPOND

2019-10-12 22:54:46.384905 -- Received packet request for message 27900, sequence 38

2019-10-12 22:54:46.388899 -- Sending packet for message 27900, sequence 38

Liite 9. Poikkeuskenaario: Laitteen B loki

TERMINAL - SANLA CLASSIC

Serial control: Port: COM8 Close COM Port Baud Rate: 115200 Debug

SanLa Control Group: 65535 Set

Messages

2019-10-12 22:54:48.680372 4236: This message contains 119 characters so it cannot be sent in a single transmission, but must be split across 4 packets.

SanLa Classic
Written by Miika Avela and Atro Lähdemäki

Send

Debug

COM Port Opened
 2019-10-12 22:54:39.096516 -- LoRaModule::onPacket
 2019-10-12 22:54:39.100520 -- Received packet for message 27900, sequence 0
 2019-10-12 22:54:39.103523 -- messaging::ActionsE::RECEIVE
 2019-10-12 22:54:39.108527 -- DownlinkBuffer::StorePacket -- Creating new DownlinkPacket.
 2019-10-12 22:54:39.113532 -- LoRaModule::onPacket -- Processing incoming packet done!
 2019-10-12 22:54:41.335287 -- LoRaModule::onPacket
 2019-10-12 22:54:41.340292 -- Received packet for message 27900, sequence 76
 2019-10-12 22:54:41.342293 -- messaging::ActionsE::RECEIVE
 2019-10-12 22:54:41.348299 -- DownlinkBuffer::StorePacket -- Existing DownlinkPacket found.
 2019-10-12 22:54:41.353303 -- LoRaModule::onPacket -- Processing incoming packet done!
 2019-10-12 22:54:43.574549 -- LoRaModule::onPacket
 2019-10-12 22:54:43.579553 -- Received packet for message 27900, sequence 114
 2019-10-12 22:54:43.581556 -- messaging::ActionsE::RECEIVE
 2019-10-12 22:54:43.587560 -- DownlinkBuffer::StorePacket -- Existing DownlinkPacket found.
 2019-10-12 22:54:43.592565 -- Requesting missing packet for message 27900, sequence 38
 2019-10-12 22:54:43.597569 -- LoRaModule::onPacket -- Processing incoming packet done!
 2019-10-12 22:54:44.110424 -- Sending packet for message 27900, sequence 38
 2019-10-12 22:54:48.645340 -- LoRaModule::onPacket
 2019-10-12 22:54:48.649344 -- Received packet for message 27900, sequence 38
 2019-10-12 22:54:48.652347 -- messaging::ActionsE::RECEIVE
 2019-10-12 22:54:48.657351 -- DownlinkBuffer::StorePacket -- Existing DownlinkPacket found.
 2019-10-12 22:54:48.663357 -- DownlinkBuffer::StorePacket -- DownlinkPacket ready to MessageStore.
 2019-10-12 22:54:48.666359 -- DownlinkBuffer::SendMessageToStore
 2019-10-12 22:54:48.669362 -- SanlaProcessor::HandleMessage
 2019-10-12 22:54:48.680372 -- 4236: This message contains 119 characters so it cannot be sent in a single transmission, but must be split across 4 packets.