Alexey Pynninen

PLC-kirjaston luominen tietokanta-hallintajärjestelmille

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Konetekniikan tutkinto-ohjelma

Insinöörityö

5.12.2021

Tiivistelmä

Tekijä:	Alexey Pynninen
Otsikko:	PLC-kirjaston luominen tietokannanhallintajärjestelmille
Sivumäärä:	50 sivua
Aika:	5.12.2021
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Konetekniikka
Ammatillinen pääaine:	Koneautomaatio
Ohjaajat:	Lead Engineer Janne Laari
	Lehtori Antti Liljaniemi

Insinöörityössä luotiin PLC-kirjasto, jonka tarkoituksena on kommunikoida kahden tietokantahallintajärjestelmän kanssa. PLC-kirjasto koostuu kahdesta funktiosta. Ensimmäisen funktion tarkoituksena oli luoda ohjelmoitavan logiikan ja MySQL-hallintajärjestelmän välisen kommunikoinnin. Toisen funktion tarkoituksena oli muodostaa yksinkertaistetun version B&R:n tarjoamasta PLC-kirjastosta, joka kommunikoi Microsoft SQL Server -hallintajärjestelmän kanssa. Hallintajärjestelmien kanssa kommunikointi sisältää tietokantaan sisäänkirjautumisen, SQL-kyselyiden lähettämisen sekä saapuvien vastauksien vastaanottamisen. PLC-kirjasto on toteutettu B&R-merkkiselle ohjelmoitavalle logiikalle.

Insinöörityön toimeksiantajana oli Etteplan Oy.

Työssä tutkittiin TCP/IP-pinon rakennetta, sekä pinon kuljetus- ja verkkokerrosprotokollien teoriaa. Tämän lisäksi työssä tutkittiin MySQL Client/Server -protokollaa. Näiden tietojen pohjalta rakennettiin PLC-kirjastoon funktio, joka kommunikoi MySQL-hallintajärjestelmän kanssa. Microsoft SQL Server -hallintajärjestelmälle olemassa oleva PLC-kirjasto yksinkertaistettiin konfiguroimalla ja suorittamalla kommunikoinnin funktioita käyttäjän puolesta.

Lopputuloksena saatiin PLC-kirjasto, joka täyttää kaikki työssä asetetut tavoitteet. Kirjasto toteutettiin helppokäyttöiseksi, jonka ansiosta hallintajärjestelmien kanssa kommunikoinnin toteuttaminen projekteissa onnistuu nopeasti.

Avainsanat: PLC, SQL, TCP/IP, MySQL, Microsoft SQL Server

Abstract

Author:	Alexey Pynninen
Title:	Creating a PLC Library for Database Management Systems
Number of Pages:	50 pages
Date:	5 December 2021
Degree:	Bachelor of Engineering
Degree Programme:	Mechanical Engineering
Professional Major:	Machine Automation
Supervisors:	Janne Laari, Lead Engineer Antti Liljaniemi, Senior Lecturer

---

In this Bachelor´s thesis, a PLC library was created to communicate with two relational database management systems. The PLC library consists of two functions. The purpose of the first function was to build communication between the programmable logic controller and the MySQL database management system. The second function aimed to create a simplified version of the PLC library provided by B&R that interacts with the Microsoft SQL Server database management system. Communication with database management systems includes logging into the database, sending SQL queries, and receiving incoming responses. The PLC library has been implemented for B&R programmable logic. The Bachelor´s thesis was commissioned by Etteplan Oy.

The structure of the TCP/IP, as well as the theory of transport and internet layer protocols, were studied. In addition, the MySQL Client/Server protocol was examined in this thesis. Based on this information, a function was built in the PLC library that communicates with the MySQL database management system. For the Microsoft SQL Server database management system, the existing PLC library was simplified by configuring and executing communication functions on behalf of the user.

As a result, a PLC library was created that meets the objectives set in the thesis. The library was implemented to be easy to use, which makes it possible to quickly communicate with database management systems in projects.

Keywords: PLC, SQL, TCP/IP, MySQL, Microsoft SQL Server

Sisällys

Lyhenteet ja käsitteet

IP: Internet protocol. Internet-protokolla.

PLC: Programmable Logic Controller. Ohjelmoitava logiikka.

ST: Structured text. Standardisoitu ohjelmointikieli.

SQL: Structured Query Language. Kyselykieli relaatiotietokannoille.

TCP: Transmission Control Protocol. Verkkotason palvelun määrittävä protokolla.

XOR: Eksklusiivinen OR-operaattori.

1 Johdanto

Relaatiotietokantoja käytetään nykypäivänä hyvin laajasti erilaisissa sovelluksissa. Sen johdosta relaatiotietokannan on muodostettava yhteys lukemattomiin sovelluksiin tiedon keräämistä, päivittämistä tai luovuttamista varten. Automaatiojärjestelmät ovat myös yksi näistä sovelluksista. Tietokannasta saadun tiedon perusteella automaatiojärjestelmä voi ohjata prosessia, suorittaa tehtäviä, sekä lähettää päivitettyä tai uutta tietoa takaisin tietokantaan.

Opinnäytetyö tehdään insinööritoimiston Etteplan Finland Oy:n toimeksiannosta. Tavoitteena on luoda PLC-kirjasto B&R-merkkiselle ohjelmoitavalle logiikalle. PLC-kirjastolla rakennetaan kommunikointia kahden relaatiotietokannan hallintajärjestelmän kanssa. Kommunikointi tulee koostumaan yhteyden muodostamisesta, SQL-kyselyn lähettämisestä sekä hallintajärjestelmältä saapuvan datan vastaanottamisesta. Hallintajärjestelmät, joiden kanssa PLC-kirjasto tulee kommunikoimaan ovat MySQL ja Microsoft SQL Server.

Vaikka MySQL-hallintajärjestelmälle on jo olemassa PLC-kirjasto, sen toteutustapa poissulkee kirjaston käytön useimmissa projekteissa, sekä luo tarpeen kehittää oman ratkaisun. Syynä siihen on se, että toimiakseen olemassa oleva kirjasto, vaatii python-skriptin suorittamisen serverin, eli asiakkaan koneella ja tietoturvasyistä todennäköisesti tämä ei sallita tehdä. Microsoft SQL Server -hallintajärjestelmälle on myös olemassa kirjasto, joka tullaan kehittämään yksinkertaiseksi ja helppokäyttöiseksi.

Yhteys MySQL-tietokantaan toteutetaan TCP-protokollan avulla. Saman kirjaston alle liitetään työn aikana kehittämä Microsoft SQL Server -kirjasto. Yhtenäinen kirjasto helpottaa ja nopeuttaa PLC-ohjelmoinnin tulevissa projekteissa, joissa on tarvetta kommunikoida tietokantojen kanssa.

2 Ohjelmoitava logiikka

Programmable Logic Controller lyhennettynä PLC tarkoittaa ohjelmoitavaa logiikkaa. Ohjelmoitavan logiikan voi verrata teollisuustietokoneeseen, se on hyvin yleinen komponentti automaatiojärjestelmissä, joka sisään- ja ulostulojen avulla ohjaa erilaisia prosesseja. Työssä luodun PLC-kirjaston toiminta on testattu B&R:n X20CP1586-mallisella logiikalla. Kyseillä logiikalla on Intel Atom 1.6 GHz -pääprosessori sekä I/O-lisäprosessori. Logiikkaan on sisäänrakennettu valmius Ethernet- ja POWERLINK-yhteyteen.

2.1 PLC-ohjelmointi

PLC-kirjasto on ohjelmoitu ja testattu Automation studio -ohjelmointiympäristössä. Se on B&R:n kehittämä ohjelmisto, joka on tarkoitettu heidän valmistamaan logiikan ohjelmointiin. Automation studio tukee kaikki IEC 61131-3 standardin mukaiset ohjelmointikielet, jotka ovat:

- Function Block Diagram
- Ladder Diagram
- Instruction List
- Sequential Function Chart
- Structured Text

Näiden lisäksi Automation studio tukee myös seuraavat ohjelmointikielet:

- Continuous Function Chart
- ANSI C
- C++. (Programming.)

Opinnäytetyö on toteutettu Structured Text ja ANSI C -ohjelmointikielillä.

PLC-kirjaston ohjelmoinnissa on hyödynnetty CASE-valintarakennetta. Valintarakenne koostuu osiin jaetusta ohjelmasta. Jokaisella osalla on oma tunniste, jonka määrittämällä ohjelma suorittaa määritetyn osan, muita osia huomiomatta. Valintarakenteessa ohjelma ohjataan seuraavaan osaan, kun tietyt ehdot ovat täyttyneet. Tämän avulla saavutetaan ohjelman hallittu eteneminen.

Monet tiedot ohjelma tallentaa listoihin, joista osa ovat kaksiulotteisia. Kaksiulotteinen lista voidaan kuvitella taulukoksi, jossa rivit vastaavat ulotteisuuksien määrän ja sarakkeet vastaavat listan sisältämiä tietoja. Taulukko 1 voidaan ilmaista listana näin: ([1, 2, 3],[4, 5, 6],[7, 8, 9]). Listan arvot alkavat nollasta näin ollen arvon 7 paikka listassa voidaan ilmaista näin: [2, 0].

Taulukko 1. Kaksiulotteinen taulukko

1	2	3
4	5	6
7	8	9

2.2 PLC-kirjastot

PLC-kirjastot helpottavat ja nopeuttavat ohjelman kirjoittamista, sekä tekevät ohjelmasta selkeämmän ja helppolukuisen. Kirjasto sisältää funktioita tai funktioblokkeja, riippuen tarvitseeko muuttujien arvoja säilyttää muistissa. Tästä lähtien opinnäytetyössä molemmista käytetään yhteistä funktio-nimeä. Funktiot suorittavat tietyn algoritmin käyttäjän antamilla muuttujilla. Esimerkkinä yksinkertaisesta funktiosta voisi olla ympyrän pinta-alan laskeminen, joka voidaan laskea $A = \pi r^2$ kaavan mukaan. Tämän funktion kutsuttaessa käyttäjä antaisi ympyrän säteen ja funktio palauttaisi ympyrän pinta-alan, jonka se on laskenut kaavan mukaan. Todellisuudessa kirjastot sekä niiden sisältämät funktiot ovat

paljon monimutkaisemmat, eivätkä rajoitu mihinkään käyttöalueeseen. Kirjasto kannattaa luoda, jos samanlaista algoritmia käytetään usein tai sen uudelleenluonti on hankalaa.

PLC-kirjaston luominen tarkoittaa algoritmien ohjelmointia funktioihin, joita myöhemmin kutsutaan ohjelmissa. Funktio voi myös hyödyntää muiden kirjastojen funktioita.

Tässä opinnäytetyössä on käytetty useita PLC-kirjastoja, monet niistä ovat valmiiksi tuotu Automation Studion puolesta ja mahdollistavat perustoimintoja kuten loogisten operaattoreiden käytön. Monia kirjastoja on lisätty kuitenkin itse. Tärkeimmät niistä opinnäytetyön kannalta ovat Etteplanin kehittämä yksinkertaistettu TCP-kirjasto, joka pohjautuu B&R:n viralliseen TCP-kirjastoon, sekä B&R:n tarjoama AsDb-kirjasto. Ensimmäisen kirjaston avulla on rakennettu kommunikointi MySQL-hallintajärjestelmän kanssa. AsDb-kirjasto toteuttaa kommunikoinnin Microsoft SQL Server -hallintajärjestelmän kanssa. Työssä yhteen kirjastoon on yhdistetty molemmat funktiot, jotka toteuttavat kommunikoinnin eri hallintajärjestelmien kanssa.

3 TCP/IP-protokolla

Internet Protocol Suite viittaa yhdistelmään monista tietoliikenneprotokolloista eli pinoon. Tämä tietoliikenneprotokollan yhdistelmä tunnetaan TCP/IP-lyhennyksellä, joka tulee pinon kahdesta tärkeimmästä protokollasta, jotka ovat Transmission Control Protocol (TCP) ja Internet Protocol (IP). Standardit kutsutaan protokolleiksi ja ne määrittävät tietoliikenteessä käytettävät toteutustavat kuten syntaksia, viestien muodon sekä kuvaavat tietokoneiden toiminnan viestien saapuessa. Protokollat mahdollistavat tiedonsiirron huolimatta siitä, minkälainen verkkolaite on käytössä. (Comer 2002: 2–6.)

TCP/IP-pino koostuu neljästä kerroksesta, jotka ovat:

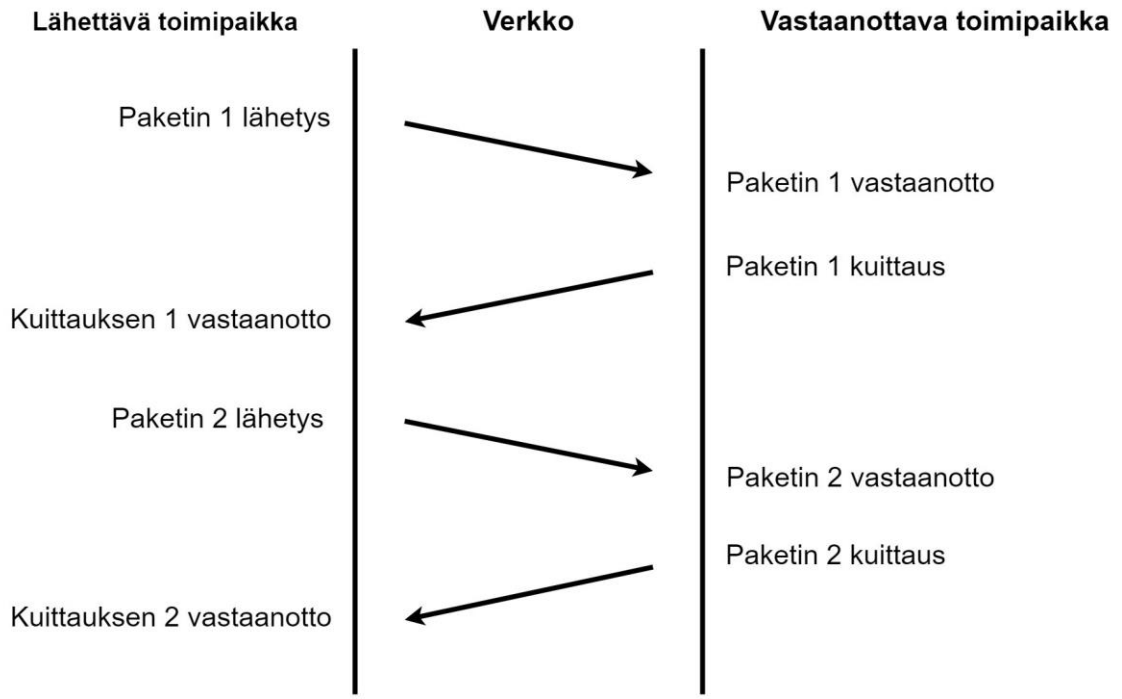
- sovelluskerros
- kuljetuskerros
- Internet-kerros
- verkkokerros (Comer 2002: 184).

Jokainen kerros on riippuvainen alla olevista kerroksista, jotka luovat perustan pakettien yhteydettömään kuljetuspalveluun.

3.1 TCP-protokolla

TCP on itsenäinen ja yleiskäyttöinen kuljetuskerroksen protokolla, joka tarjoaa luotettavan virtaorientoituneen kuljetuspalvelun eli tiedonsiirron.

Tiedonsiirtoverkot alimmalla tasolla eivät pysty tarjoamaan luotettavaa kuljetuspalvelua. Paketit voivat kadota tai vahingoittua, jos verkon laitteet ovat epäkunnossa tai verkko on kuormitettu liikaa. Verkoissa, joissa reititys on dynaamista, paketit voivat saapua satunnaisessa järjestyksessä. Virtaorientoitunut ja luotettava kuljetus takaa sen, että kaikki paketit saapuvat käyttäjälle perille oikeassa järjestyksessä. TCP-protokolla saavuttaa tämän käyttämällä positiiviset kuittaukset ja uudelleenlähetykset -nimistä menetelmää. Tämä menetelmä on esitetty kuvassa 1 ja perustuu siihen, että vastaanottaja lähettää kuittauksen (acknowledgement, ACK) lähettäjälle paketin vastaanotettuaan.



Kuva 1. Positiiviset kuittaukset ja uudelleenlähetykset menetelmä (Comer 2002: 212)

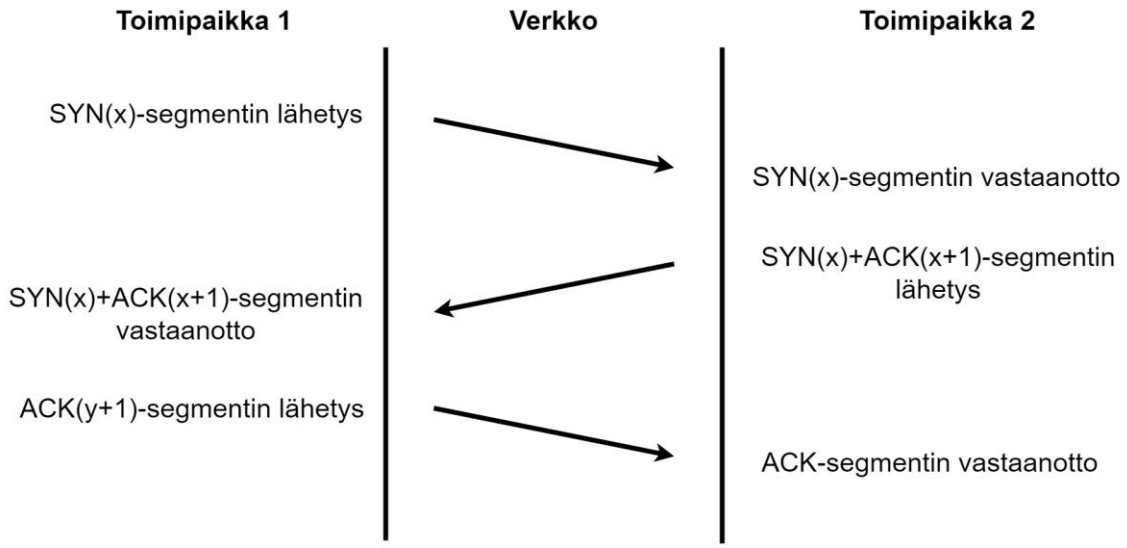
Paketti lähetetään uudestaan, jos kuittaus ei saapunut lähettäjälle tietyn ajan puitteissa.

Seuraavan paketin lähettäminen vasta silloin, kun edellisen paketin kuittaus on saapunut, ei ole tehokasta. Verkon kaistanleveyden tehokkaamman käytön saavuttamiseksi käytetään protokollaa, joka tunnetaan nimellä liukuva ikkuna. Liukuva ikkuna -protokolla sallii useiden pakettien lähettämisen toistensa perään odottamatta kuittauksia. Protokolla asettaa ikkunalle tietyn koon, joka määrää kuinka monta pakettia voi yhdellä kerralla lähettää. Ikkunan ulkopuolella odottavat paketit lähetetään, kun ikkunapaikkoja vapautuu eli vastaanotetaan kuittaukset jo lähetetyistä paketeista. Protokolla pitää muistissa jokaisen kuittaamattoman paketin ajastimen uudelleenlähetystä varten, jos paketti katoaa. (Comer 2002: 209–214.)

TCP-yhteys määritetään kahdella päätepisteellä, jotka koostuvat IP-osoitteesta ja isäntälaitteen portista. Yhteys yksilöidään päätepisteiden mukaan, joka tarkoittaa sen, että sama IP-osoite ja portti voivat jakaa samanaikaisesti useita yhteyksiä. TCP-protokolla edellyttää, että molemmat päätepisteet hyväksyvät yhteyden muodostamisen, jonka jälkeen TCP-liikennettä voidaan kuljettaa yhteisverkossa. (Comer 2002: 217–218.)

Segmentti on siirtoyksikkö, jota TCP-ohjelmistot lähettävät toisilleen. Lähettämällä segmenttejä luodaan ja suljetaan yhteydet sekä kuljetetaan dataa, kuittaukset ja ikkunan kokoa koskevat ilmoitukset. Segmentti koostuu otsikko- ja dataosista. TCP-otsikko tunnetaan myös header-nimellä ja se sisältää tunniste- ja ohjaustiedot. Dataosa sisältää sovelluksen lähettämän datan, joka myös voi koostua otsikosta ja datasta, riippuen sovelluksen protokollasta. (Comer 2002: 221.)

Yhteyden muodostamiseen TCP-protokolla käyttää kolmivaiheista kättelyä, joka takaa osapuolien valmiutta tiedonsiirtoon. Kättely tapahtuu kuvan 2 mukaisesti.

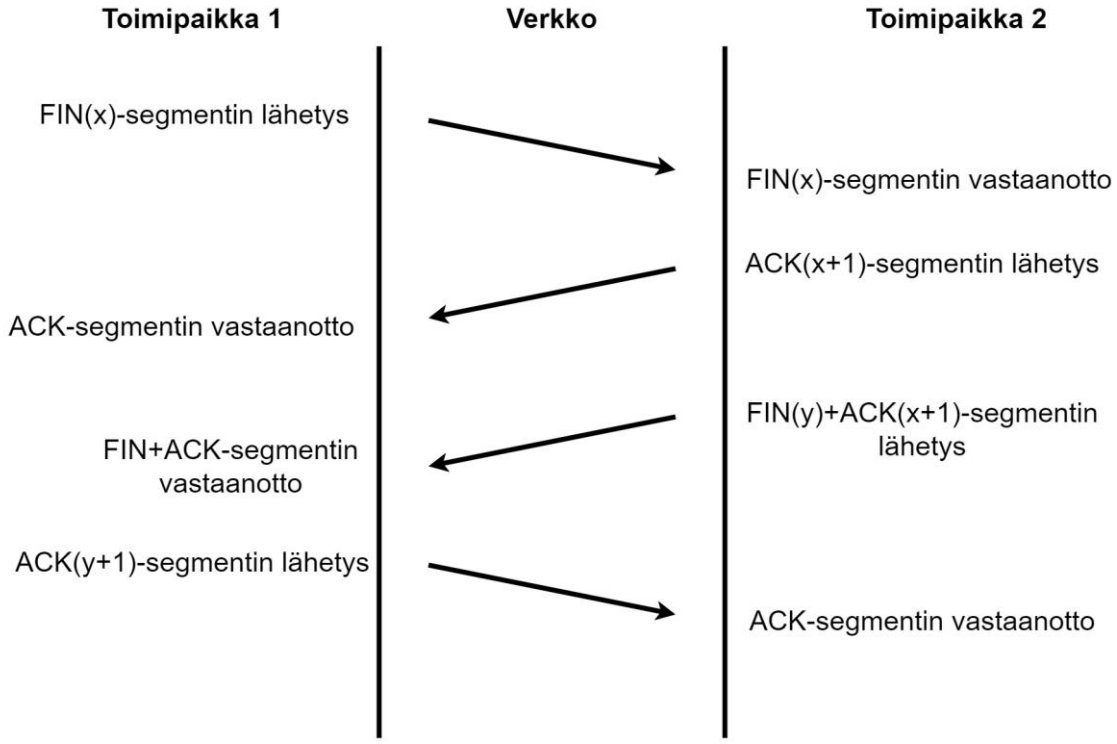


Kuva 2. TCP-protokollan kolmivaiheinen kättely (Comer 2002: 238)

Tietokone A, joka aloittaa kättelyn luo satunnaislukugeneraattorilla järjestysnumeron x, jonka se liittää ensimmäiseen SYN-segmenttiin (lyhenne sanasta

synchronization eli synkronointi) ja lähettää sen tietokoneelle B. Tietokoneen B vastanotettua SYN-segmentin, luo se oman järjestysnumeron y satunnaislukugeneraattorilla, sekä kasvattaa vastaanotetun järjestysnumeron x yhdellä. Tämän vaiheen jälkeen tietokone B lähettää takaisin muodostetun SYN(y)+ACK(x+1)-segmentin. Kättelyn viimeisenä vaiheena tietokone A kasvattaa vastaanotetun järjestysnumeron y yhdellä ja kuittaa sen lähettämällä ACK(y+1)-segmentin. Näiden vaiheiden jälkeen TCP-yhteys on muodostettu. (Comer 2002: 237–238.)

Sovellukset voivat katkaista TCP-yhteyden hallitusti. Yhteyden suljettaessa käytetään muunnettua kolmivaiheista kättelyä kuvan 3 mukaisesti. TCP-yhteys on kaksisuuntainen ja yhteydet muodostuvat kahdesta itsenäisestä virrasta, siksi yhteydet suljetaan jokaiseen suuntaan erikseen.



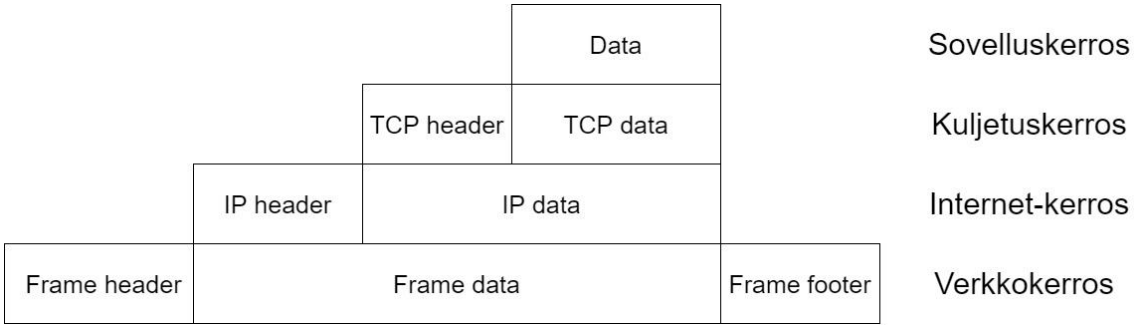
Kuva 3. TCP-yhteyden sulkeminen (Comer 2002: 240)

Kun sovellus A haluaa sulkea yhteyden se lähettää FIN(x)-segmentin, johon vastauksena sovellukselta B saa kuittauksen ACK(x+1)-segmentin muodossa.

Nyt yhteys yhteen suuntaan on suljettu. Yhteys toiseen suuntaan suljetaan lähettämällä FIN(y)+ACK(x+1)-segmentin tietokoneelle A, joka aloitti yhteyden sulkemisen. Kuittauksena saapuu ACK(y+1)-segmentti, jonka jälkeen TCP-yhteys on kokonaan suljettu.

3.2 IP-protokolla

IP-protokolla (Internet Protocol) on yhteisverkkojen toiseksi tärkeimpiä protokollia. TCP/IP-pinossa IP-protokolla on verkkokerroksessa, eli yksi taso TCP-protokollaa alempana. Kuvassa 4 on esitetty TCP/IP-pino sekä tasojen riippuvuudet toisistaan.



Kuva 4. TCP/IP-pino

Kolme tärkeää asiaa, joita IP-protokolla määrittää:

1. IP-protokolla määrittää TCP/IP-verkossa tapahtuvan kuljetuksen perusmuodon eli kaiken yhteisverkossa siirrettävän datan muodon.

2. IP-ohjelmat valitsevat reitin, jota pitkin tiedot kuljetetaan.

3. IP sisältää joukon sääntöjä, jotka määrittävät kuinka asemat ja reitittimet käsittelevät paketteja, missä tilanteessa tulee luoda virhesanomia ja olosuhteet, joissa paketit tulee hylätä. (Comer 2002: 95,97.)

IP-verkon perusyksikkö on IP-tietosähke, joka koostuu otsikko- ja dataosasta. Otsikko sisältää muun muassa lähteen ja kohteen IP-osoitteet, otsikon pituuden, kokonaispituuden sekä prioriteetin. (Comer 2002: 113.)

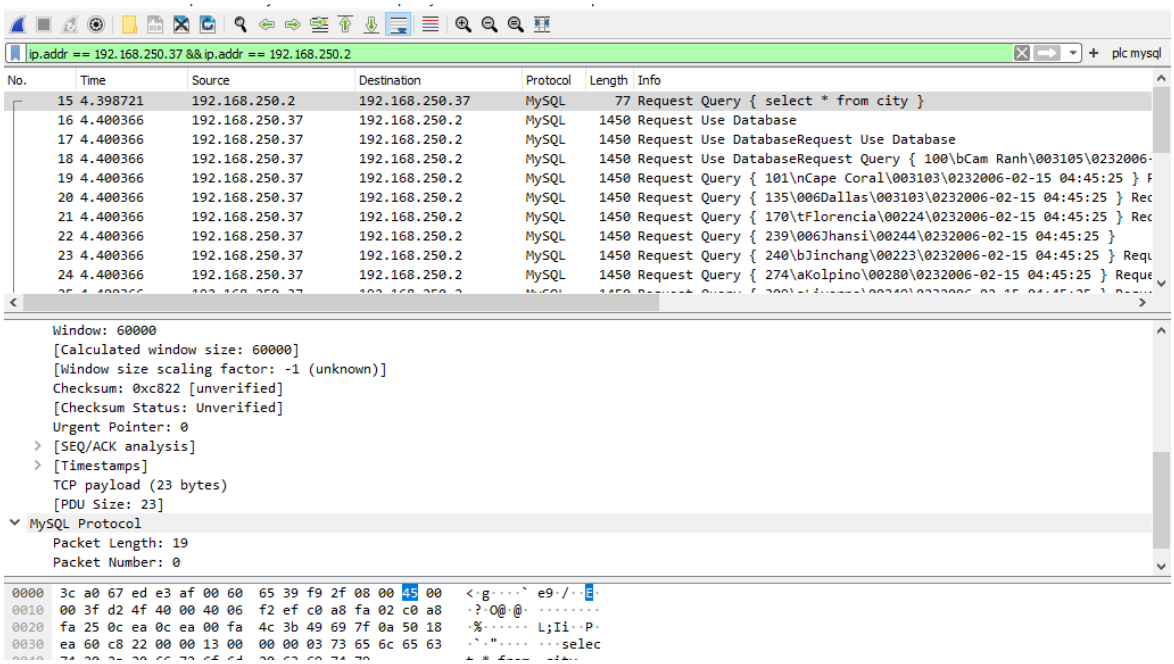
Kohteen IP-osoitteen perusteella tehdään päätös mihin seuraava tietosähke lähetetään. Jos lähettäjä ja vastaanottaja ovat samassa verkossa tietosähke voidaan lähettää suoraan. Tilanteessa, jossa suoraan lähettäminen ei onnistu, tietosähkeet lähetetään epäsuorasti lähimmälle reitittimelle. Yhteisverkon kautta tietosähke kulkee reitittimeltä reitittimelle, kunnes se voidaan lähettää suoraan kohteeseen yhden fyysisen verkon kautta. (Comer 2002: 125.)

3.3 Socket-ohjelmointirajapinta

Socket-ohjelmointirajapinta tarjoaa kirjastorutiineja, joiden avulla sovellukset kommunikoivat TCP/IP-verkossa. Socket-termiä voidaan kuvata mekanismina, joka muodostaa kommunikoinnin päätepisteen. Yhteyden muodostamista varten sovellus pyytää järjestelmän luomaan socketin. Sovelluksen on määritettävä järjestelmälle verkko-osoitteen, porttinumeron sekä käytettävän protokollaperheen, johon yhteys halutaan luoda. Järjestelmä palauttaa vastauksena kokonaisluvun, jota sovellus käyttää viitatessaan luotuun sockettiin. Yhteyden muodostettua sovellus voi käyttää read- ja write-komentoja tietojen lukemista ja kirjoittamista varten. TCP-socketin avulla luotu PLC-kirjasto vaihtelee tietoja hallintajärjestelmien kanssa. (Comer 2002: 413–416; What is a TCP/IP Socket Connection?.)

3.4 Wireshark-ohjelma

Wireshark on avoimeen lähdekoodiin perustuva työkalu verkkopakettien analysointiin. Ohjelman avulla kaapataan samassa verkossa olevia verkkopaketteja. Kaapatut verkkopaketit voidaan suodattaa haluamalla tavalla ja analysoida datapakettien sisällön. Tässä työssä Wireshark-ohjelman avulla on havainnollistettu PLC:n ja MySQL-hallintajärjestelmän datanvaihtoa, sekä hyödynnetty virheidenetsinnässä ohjelman rakentamisen ja testauksen vaiheissa.



Kuva 5. Wireshark-ohjelma

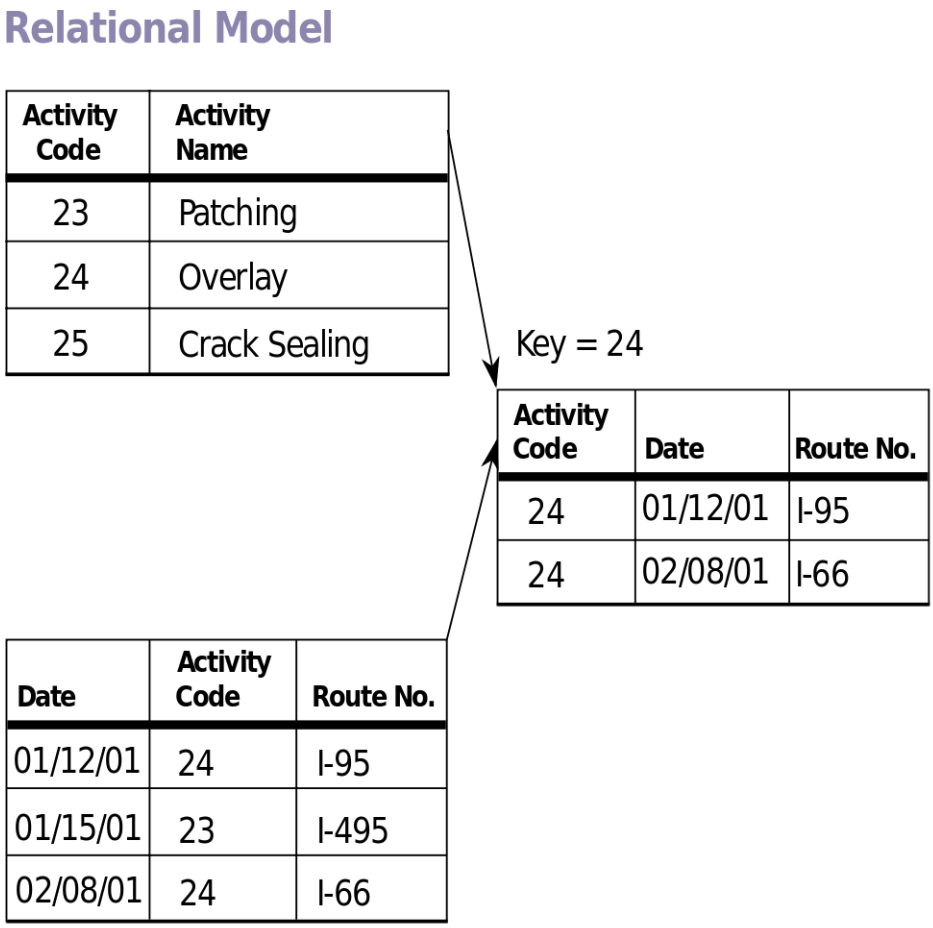
Ohjelman näkymän voi jakaa neljään osaan. Ylin osa (kuvassa 5 korostettu vihreällä) on suodatin, jonka avulla löydetään datapakettien joukosta ne, joita halutaan tutkia. Kuvan 5 esimerkkikaappauksessa paketit ovat suodatettu kahdella IP-osoitteella, eli tarkasteltiin kahden laitteen välistä kommunikaatiota. Suodattimen jälkeen on lista kaapatuista paketeista. Listan jälkeen on ikkuna, joka näyttää valitun paketin sisällön ohjelman tulkitussa muodossa. Ohjelma tunnistaa minkä protokollan paketti on kyseessä ja kääntää sen luettavaan muotoon. Viimeisenä on valitun paketin raakadata eli käsittelemätön data, jonka ohjelma näyttää joko binaareina tai heksadesimaalilukuina. Viimeisen vaihtoehdon on huomattavasti helpompi lukea. Vaikka ohjelma osaa tulkita MySQL-protokollan, ohjelman toteutuksen aikana on huomattu, ettei kaikki paketit käänny oikein. Tämän takia paketteja työssä on tarkasteltu ainoastaan heksadesimaalilukuina.

4 Relaatiotietokannat

Tietokantojen nopeus, tarkkuus ja huolellisuus tekevät niistä korvaamattoman työkalun nykypäivän tiedonvaihdossa. Tietokannat antavat käyttäjille ja sovelluksille nopean pääsyn tietojen lukemiseen ja kirjoittamiseen. Tieto voidaan kerätä, ilmaista ja analysoida räätälöidysti.

4.1 Relaatiotietokantojen rakenne

Relaatiotietokannat perustuvat relaatiomalliin. Kaikki tieto relaatiokannoissa on tallennettu tauluina, jossa on sarakkeita ja rivejä. Jokaisessa taulussa tunnisteena on uniikki perusavain, joka yksilöi rivin. Rivin loput tiedot ovat viiteavaimia. Perusavain voi koostua monesta sarakkeesta ja esiintyä eri tauluissa. Viiteavaimet viittaavat perusavaimeen, jolloin muodostuu yhteys. Perusavaimella voi olla yhteys moneen viiteavaimeen, mutta viiteavaimella ainoastaan yhteen perusavaimeen. (Hovi: 5–6.)



Kuva 6. Esimerkki relaatiomallista (Relaatiotietokanta)

Kuvassa 6 on esimerkki relaatiomallista. Ylemmässä taulussa työn koodi (activity code) on perusavaimena ja työn nimi on viiteavaimena. Alemmassa taulussa perusavaimena on päivämäärä ja viiteavaimina on työn koodi ja reitti. SQL-kyselyllä voidaan hakea tiedot, milloin ja mitä reittiä työtä koodilla 24 on tehty, josta tulokseksi saadaan keskimmäisen taulukon.

4.2 SQL

Structured Query Language lyhennettynä SQL on standardisoitu ohjelmointikieli (ISO/IEC 9075), jonka avulla hallintajärjestelmät kommunikoivat relaatiotietokantojen kanssa (Kelechava). Yleisimmät SQL-operaattorit sekä niiden sisältämät komennot ovat:

- Tietojenmäärittely operaattori (Data Definition Language)
  - CREATE – luo tietokannan objektin
  - ALTER – muokkaa objektin
  - DROP – poistaa objektin
- Tietojenkäsittely operaattori (Data Manipulation Language)
  - SELECT – valitsee tietoja, jotka vastaavat määritettyjä ehtoja
  - INSERT – lisää uusia tietoja
  - UPDATE – muokkaa olemassa olevat tiedot
  - DELETE – poistaa tiedot
- Tiedonsaantioperaattori (Data Control Language)
  - GRANT – antaa käyttäjälle oikeudet tiettyihin toimintoihin
  - REVOKE – peruuttaa aiemmin myönnetyt oikeudet. (MySQL Glossary.)

4.3 Hallintajärjestelmät

Hallintajärjestelmä on ohjelmistojärjestelmä, jonka avulla käyttäjät voivat määrittää, luoda, ylläpitää ja valvoa pääsyä tietokantaan. Tässä työssä PLC-kirjasto on toteutettu kommunikoimaan MySQL- ja Microsoft SQL Server -hallintajärjestelmille. MySQL Workbench ja Microsoft SQL Server Management Studio ovat visuaaliset työkalut, joissa toimitaan hallintajärjestelmien kanssa.

5 MySQL Client/Server -protokolla

Ennen MySQL-funktion varsinaista suunnittelua ja toteutusta on olennaista tuntea MySQL-protokolla. MySQL-protokolla määrittää asiakkaan ja palvelimen välistä kommunikointia.

5.1 Protokollan perusteet

Protokolla käyttää useita erilaisia tietotyyppejä. Pohjimmiltaan tietotyypit ovat joko kokonaislukuja (integers) tai merkkijonoja (strings). Kaikki MySQL:n käyttämät tietotyypit ovat esitetty selityksineen taulukossa 2. (Integer Types; String types.)

Taulukko 2. MySQL-protokollan datatyypit (Integer Types; String types)

Datatyyppi	**Lyhenne**	**Kuvaus**
Fixed-Length integer	int<n>	Kiinteäpituinen kokonaisluku, joka pitää arvonsa tavusarjassa, vähiten merkitsevä tavu ensin.
Length-encoded integer	int<lenenc>	Pituuskoodattu kokonaisluku, joka koostuu 1, 3, 4 tai 9 tavusta, numeerisesta arvostaan riippuen.
Fixed-Length string	string<fix>	Kiinteäpituinen merkkijono.
Null-Terminated string	string<NULL>	Merkkijono, joka päättyy 00 tavuun.
Variable-Length string	string<VAR>	Merkkijono, jonka pituus määräytyy toisen kentän mukaan tai se lasketaan ajon aikana.
Length-Encoded string	str<lenenc>	Merkkijono, jonka pituus on määritetty etuliitteessä. Etuliite on pituuskoodattu kokonaisluku.
Rest Of Packet string	string<EOF>	Merkkijonon ollessa paketin viimeinen komponentti, sen pituus lasketaan vähentämällä paketin kokonaispituus nykyisestä sijainnista.

MySQL-protokolla määrittää tarkkaan pakettien sisällön sekä koon. Paketit jaetaan 2^24 tavuisiksi, jos paketin koko ylittää 16 megatavua. Jokainen paketti alkaa neljän tavun pituisella otsikolla, joka koostuu:

- int<3> datatyypistä, joka ilmoittaa datapaketin kokonaispituuden, lukuun ottamatta nelitavuisen otsikon
- int<1> datatyypistä, joka kertoo paketin sekvenssinumeron.

Vastauksena useimmille komennoille, joita lähetetään palvelimelle, palvelin palauttaa yhden seuraavista paketeista:

- OK-paketti
- ERR-paketti
- EOF-paketti.

OK- tai EOF-paketti on palvelimelta saapuva paketti, joka ilmoittaa käyttäjälle komennon onnistuneesta suorittamisesta. Sääntönä kumman paketin palvelin lähettää on:

- OK-paketti (ensimmäinen tavu otsikon jälkeen on 00) lähetetään, jos paketin pituus on >7,
- EOF-paketti (ensimmäinen tavu otsikon jälkeen on fe) lähetetään, jos paketin pituus on <9. (OK_Packet.)

On muistettavaa, että TCP-segmentti voi sisältää useita MySQL-protokollan paketteja, jossa jokaisella paketilla on oma otsikko-osa, joka ilmoittaa juuri kyseisen paketin pituuden ja sekvenssin. Taulukossa 3 on ilmaistu mistä OK-paketti koostuu.

Taulukko 3. OK-paketin data (OK_Packet)

Datatyyppi	**Nimi**	**Kuvaus**
int<1>	header	0x00 tai 0xFE OK-paketin otsikko
int<lenenc>	affected_rows	vaikutuksen alaisia rivejä
int<lenenc>	last_insert_id	viimeinen insert-id
if capabilities & CLIENT_PROTOCOL_41 {		
int<2>	status_flags	tilalippu
int<2>	warnings	varoituksien määrä
} else if capabilities & CLIENT_TRANSACTIONS {		
int<2>	status_flags	tilalippu
}		
if capabilities & CLIENT_SESSION_TRACK		
string<lenenc>	info	ihmisen luettavissa tilatieto
if status_flags & SERVER_SESSION_STATE_CHANGED {		
string<lenenc>	session state info	istunnon tilatieto
}		
} else {		
string<EOF>	info	ihmisen luettavissa tilatieto
}		

Palvelimelta lähtevä paketti voi vaihdella vastaanottajasta riippuen. Tämä johtuu taulukossa esitetyistä if-ehdoista, jotka muuttavat paketin sisältöä riippuen siitä mitkä ominaisuusliput vastaanottajalla on käytössä.

Taulukosta 4 nähdään että, EOF-paketti voi sisältää vain vähän tietoa verrattuna OK-pakettiin, kuten yllä mainitusta säännöstä on todettu. (OK_Packet; EOF_Packet.)

Taulukko 4. EOF-paketti (EOF_Packet)

Datatyyppi	**Nimi**	**Kuvaus**
int<1>	header	fe, EOF-paketin otsikko
if capabilities & CLIENT_PROTOCOL_41 {		
int<2>	warnings	varoituksien määrä
int<2>	status_flags	palvelimen tilaliput

Otetaan esimerkin vuoksi EOF-paketti heksadesimaalilukuina. EOF-paketti ilman varoituksia ja AUTOCOMMIT (arvo 2) tilalippu käytössä näyttäisi tältä:

```
05 00 00 05 fe 00 00 02 00
```

Ensimmäinen osa paketista on otsikko, joka on nelitavuinen eli 05 00 00 05. Se kertoo paketin koon (pituuden), joka on 5 tavua (05 heksadesimaalilukuna on 5) sekä sekvenssinumeron, joka on myös 5. EOF-paketin varsinainen data on:

```
fe 00 00 02 00.
```

Ensimmäisestä tavusta (fe) voidaan päätellä, että kyseessä on EOF-paketti. Seuraavat kaksi tavua ilmoittavat varoituksista, koska niitä ei kuitenkaan ole, tavujen arvot ovat 00. Loput kaksi tavua ilmoittavat mitkä tilaliput ovat käytössä. Tässä esimerkissä käytössä oleva tilalippu on 02 00 (AUTOCOMMIT).

ERR-paketti ilmoittaa käyttäjälle, että on tapahtunut jokin virhe. Paketti sisältää sekä virhekoodin että tekstitiedon virheestä. Taulukossa 5 nähdään mistä virhepaketti koostuu.

Taulukko 5. ERR-paketti (ERR_Packet)

Tyyppi	**Nimi**	**Kuvaus**
int<1>	header	ff, ERR-paketin otsikko
int<2>	error_code	virhekoodi
if capabilities & CLIENT_PROTOCOL_41 {		
string[1]	sql_state_marker	# merkki SQL-tilasta
string[5]	sql_state	SQL-tila
}		
string<EOF>	error_message	ihmisen luettavissa virheilmoitus

Otetaan toiseksi esimerkiksi isomman paketin eli virhepaketin. Heksadesimaaleina se voisi näyttää tältä (luetaan vasemmalta oikealle ja ylhäältä alas):

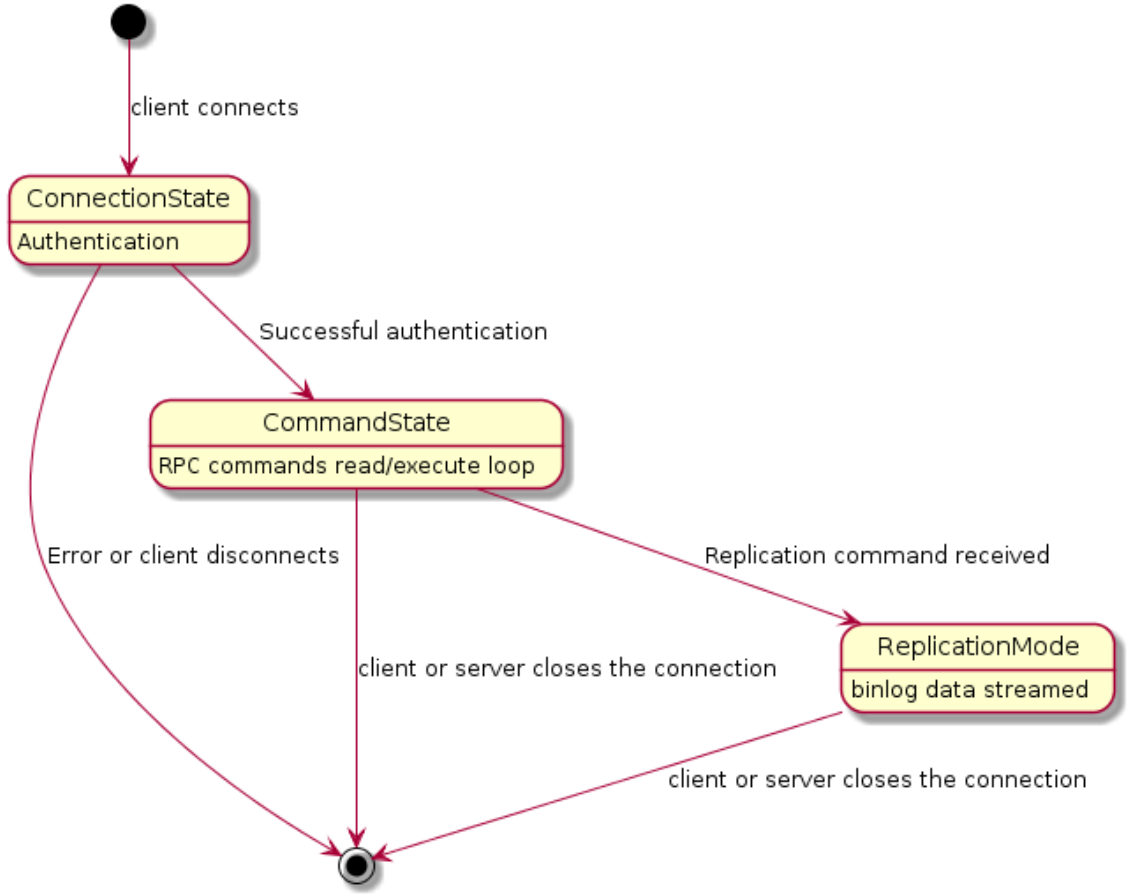
```
17 00 00 01 ff 48 04 23    48 59 30 30 30 4e 6f 20
74 61 62 6c 65 73 20 75    73 65 64
```

(ERR_Packet).

Nelitavuinen otsikko 17 00 00 01 kertoo, että paketti on 23 (heksadesimaali 17 lukuna on 23) tavun pituinen ja sekvenssinumero on 1. Virhekoodin saamiseksi tavujen paikkoja on vaihdettava eli 04 48, josta saadaan käännettynä luvuksi 1096 virhekoodin. Seuraavaksi on #-merkki (tavu 23) sekä HY000-merkkijono (tavut 48, 59, 30, 30 ja 30). Taulukosta 5 todetaan seuraavan datatyypin olevan string<EOF> eli loput tavut ovat jono peräkkäisiä merkkejä. Merkkijono sisältää ihmisen luettavissa olevan virhetekstin. Käännettynä heksadesimaaleista merkkijonoksi muodostuu ”No tables used” -teksti.

5.2 Yhteyden elinkaari

Kuvasta 7 nähdään, että palvelimen ja asiakkaan välinen yhteyden elinkaari koostuu kolmesta vaiheesta.



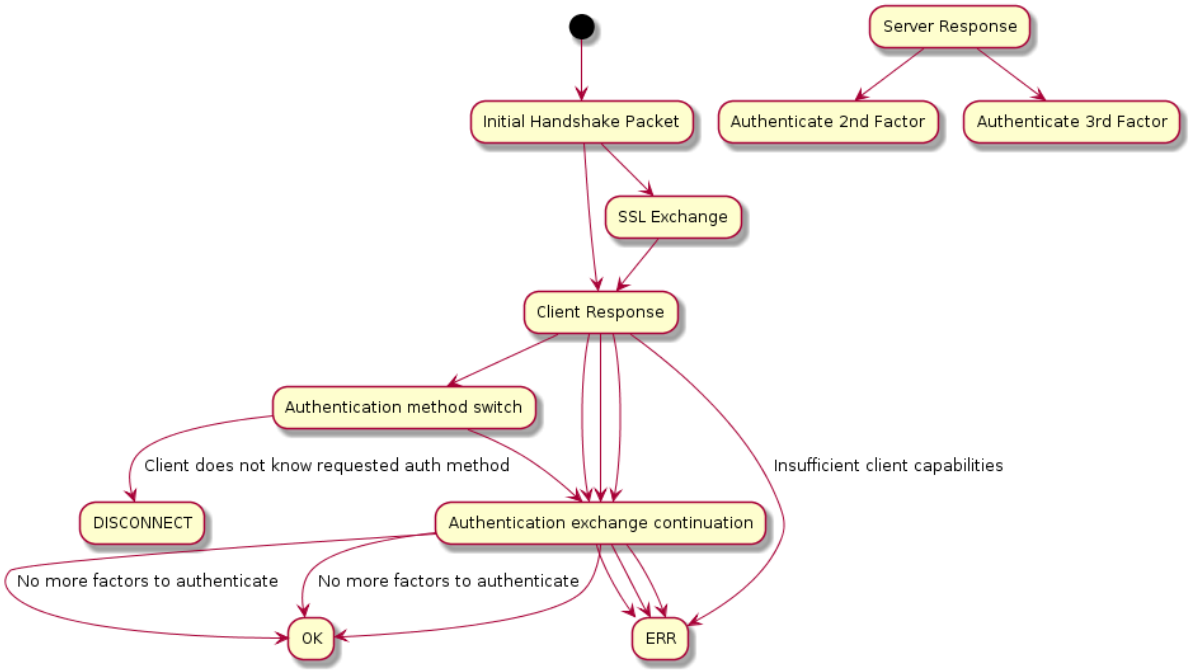
Kuva 7. Yhteyden elinkaari (Connection Lifecycle)

Ensimmäisenä vaiheena on yhteyden muodostaminen, jossa tapahtuu sisäänkirjautuminen hallintajärjestelmään. Toiseen eli komentovaiheeseen käyttäjä pääsee onnistuneella sisäänkirjautumisella. Käyttäjä pysyy komentovaiheessa, kunnes yhteys suljetaan tai lähetetään Replication-komento. Replication-komento sekä siihen liittyvää kolmannetta vaihetta ei avata tarkemmin, koska kyseinen vaihe jäi työn ulkopuolelle. (Connection Lifecycle.)

5.2.1 Yhteyden muodostamisen vaihe

Kuvasta 8 nähdään, että yhteyden muodostamisen vaihe koostuu monesta alavaiheesta. Työn rajauksen takia yhteys muodostetaan aina tiettyjen vaiheiden kautta seuraavassa järjestyksessä:

1. Sisäänkirjautumispyyntö hallintajärjestelmältä (Initial Handshake Packet),
2. Sisäänkirjautumistietojen lähettäminen (Client Response),
3. Sisäänkirjautumistietojen tarkistus (Authentication exchange continuation),
4. Hallintajärjestelmä ilmoittaa onnistuneesta kirjautumisesta tai virheestä (OK/ERR).



Kuva 8. Yhteyden muodostamisen vaihe (Connection Phase)

Työ on rajoitettu tekemällä olettamuksen, että hallintajärjestelmän sisäänkirjautumismetodi on aina Native Password. Rajaus perustuu siihen, että MySQL:n

edellinen (5.7) versio käyttää mainittua sisäänkirjautumismetodia oletuksena. Tämän lisäksi nykyiseen MySQL-versioon (8.0) voi luoda käyttäjän, joka käyttäisi kyseistä kirjautumismetodia. Rajaus poistaa tarpeen käsitellä Authentication method switch -vaihetta, sekä huomioida muita sisäänkirjautumismetodeja.

Avaamalla TCP-socketin MySQL-palvelimeen, käyttäjälle lähetetään HandshakeV10-protokollan paketti, joka sisältää tiedot palvelimesta sekä datan sisäänkirjautumista varten. Noudattaen HandshakeResponse41-protokollaa, sekä käyttämällä vastaanotetun paketin tiedot, kootaan vastauspaketti, jolla kirjaudutaan hallintajärjestelmään. Mainitut sisäänkirjautumiseen liittyvät protokollat selostetaan auki kappaleessa kuusi.

5.2.2 Komentovaihe

Onnistuneen sisäänkirjautumisen jälkeen palvelin päästää käyttäjän komentovaiheeseen. Tässä vaiheessa asiakas suorittaa SQL-kyselyitä. Taulukossa 6 Text-protokollan COM_QUERY-komentoa käyttäen ohjelma välittää kyselyn palvelimelle.

Taulukko 6. Kysely komento (COM_QUERY)

Datatyyppi	**Nimi**	**Kuvaus**
int<1>	komento	0x03: COM_QUERY
string<EOF>	kysely	suoritettava SQL-kyselyn teksti

Toinen Text-protokollan komento, jonka luotu PLC-kirjasto tukee, on COM_QUIT. Sen avulla käyttäjä sulkee yhteyden palvelimeen. Paketin rakenne on esitetty taulukossa 7.

Taulukko 7. Yhteyden sulkemisen komento (COM_QUIT)

Datatyyppi	**Nimi**	**Kuvaus**
int<1>	komento	0x01: COM_QUIT

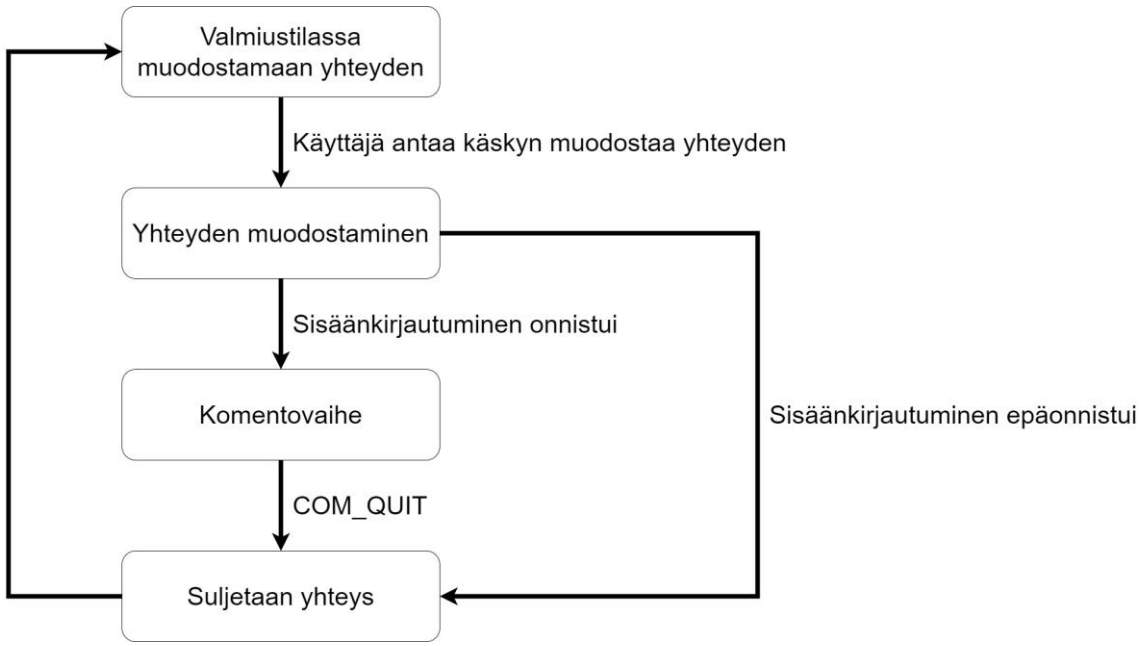
Palvelimen muut Text-protokollan komennot jäivät toteuttamatta työn aiheen ulkopuolelle menemisen ja harvan käytön takia. Työn ulkopuolelle jäävien komentojen avulla voidaan muun muassa saada palvelimelta statistiikkaa (COM_STATISTICS) ja sammuttaa palvelimen (COM_SHUTDOWN).

6 PLC-kirjaston luominen

PLC-kirjasto on toteutettu kahdesta funktiosta. Ensimmäinen funktio käsittelee MySQL-hallintajärjestelmän kanssa kommunikoinnin ja toinen funktio Microsoft SQL Server -hallintajärjestelmän kanssa. Molemmissa kirjaston funktioissa on käytetty CASE-valintarakennetta, jonka avulla ohjelma siirtyy vaiheiden välillä.

6.1 MySQL-funktion suunnittelu

Ohjelma on suunniteltu koostuvan neljästä päävaiheesta kuvan 9 mukaisesti. Jokainen päävaihe sisältää monia alavaiheita.

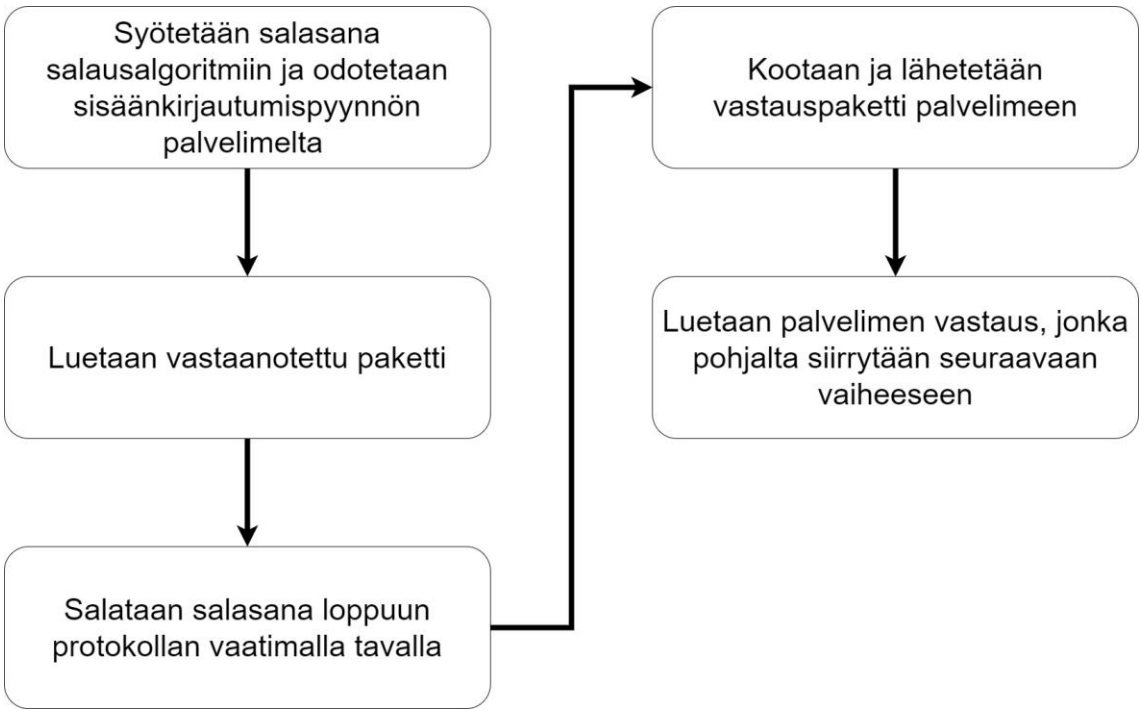


Kuva 9. MySQL-kirjaston päävaiheet

Ohjelma alkaa valmiustilasta, jossa odotetaan käyttäjän käskyä muodostamaan MySQL-palvelimeen yhteyttä. Käskyn vastaanotettua ohjelma yrittää kirjautua käyttäjän annetuilla tiedoilla. Onnistuneen kirjautumisen jälkeen ohjelma siirtyy komentovaiheeseen. Sisäänkirjautumisen epäonnistuessa virheestä ilmoitetaan käyttäjälle ja yhteys suljetaan, jonka jälkeen palataan valmiustilaan. Komentovaiheessa suoritetaan SQL-kyselyitä sekä välitetään palvelimelta

saapuvaa tietoa käyttäjälle. Käyttäjä voi pyytää sulkea yhteyden komentovaiheessa, jolloin ohjelma siirtyy kyseistä toimintoa suorittavaan vaiheeseen, jonka jälkeen ohjelma palaa valmiustilaan.

Ohjelma avaa TCP-socketin ja siirtyy muodostamaan yhteyden palvelimeen käyttäjän käskystä. Yhteyden muodostamisen vaihe on esitetty vuokaaviona kuvassa 10.

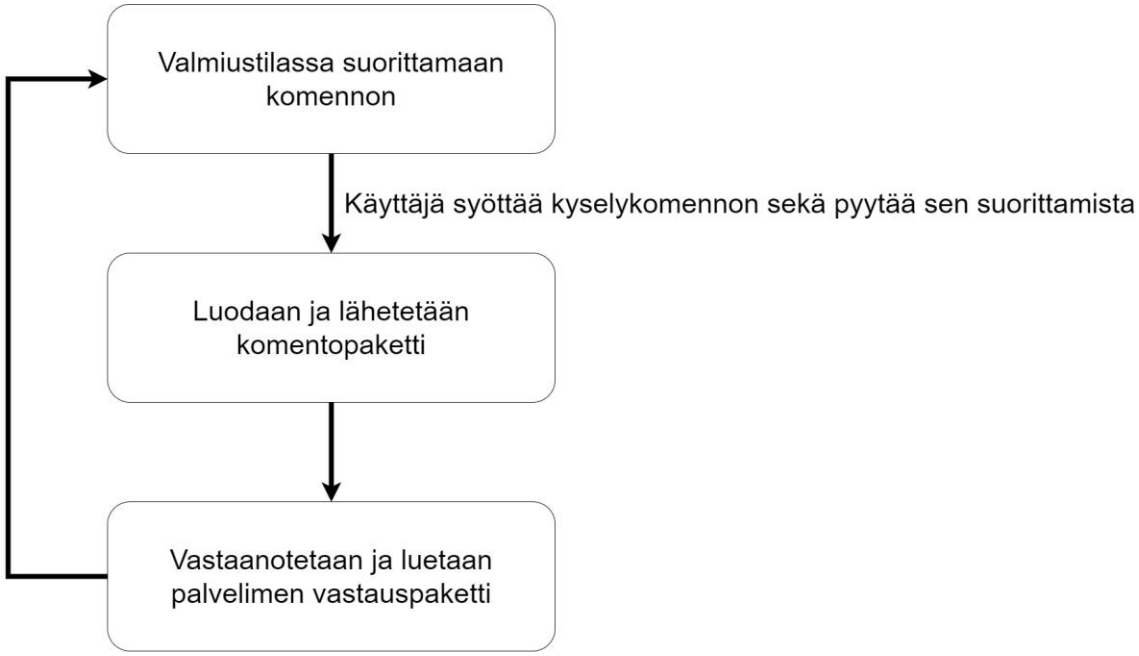


Kuva 10. Yhteyden muodostamisen vaihe

Yhteyden muodostamisen ohjelma toteuttaa viidessä vaiheessa. Ensiksi ohjelma toteuttaa käyttäjän salasanan esisalauksen, jonka jälkeen odottaa sisäänkirjautumispyynnön saapumista palvelimelta. Toisessa vaiheessa ohjelma lukee ja ottaa talteen tarvittavan tiedon palvelimen lähettämästä paketista. Tämän jälkeen ohjelma suorittaa salasanan lopullisen salauksen. Neljännessä vaiheessa ohjelma muodostaa ja lähettää vastauspaketin palvelimelle. Viimeisessä vaiheessa ohjelma vastaanottaa ja lukee palvelimen vastauspaketin, joka sisältää tiedon, onnistuiko sisäänkirjautuminen vai eikö.

Salasanan salausalgoritmin toteuttaminen on ollut työn haastavin osa. Ohjelman ensimmäinen versio on vaatinut, että salausalgoritmin alkuosa toteutetaan ennen palvelimen sisäänkirjautumispaketin vastaanottoa. Syynä siihen oli mahdollisuus ylittää palvelimen asettaman ajan, jossa vastauspaketin pitää saapua. Ohjelman viimeinen versio toteuttaa salauksen nopeasti ja tarve esisalaukselle poistui. Tästä huolimatta esisalauksen vaihe säilyi, koska käytännön kannalta ei ole merkitystä suoritetaanko salaus osissa vai kerralla. Salaukseen liittyvät yksityiskohdat käsitellään luvussa 6.2.

Onnistuneen sisäänkirjautumisen jälkeen ohjelma siirtyy komentovaiheeseen, joka on esitetty vuokaaviona kuvassa 11.



Kuva 11. Komentovaihe

Komentovaiheessa on valmiustila, jossa odotetaan käyttäjän syöttämää SQL-kyselyä. Kyselyn syötettyä ohjelma siirtyy rakentamaan protokollan mukaisen paketin, jonka se lähettää palvelimeen. Palvelimelta vastaanotettu vastauspaketti luetaan ja välitetään käyttäjälle luettavassa muodossa.

Vaiheessa, jossa yhteys suljetaan ohjelma lähettää vastaavan komennon palvelimelle, sekä alustaa muuttujat uuden yhteyden muodostamista varten. Yhteyden suljettua ohjelma palaa valmiustilaan muodostamaan yhteyden.

6.2 MySQL-funktion toteutus

Ohjelma on toteutettu ensin tavallisena PLC-ohjelmana testauksen helpottamiseksi, jonka jälkeen valmis ohjelma on siirretty PLC-kirjaston funktioon.

6.2.1 Yhteyden muodostamisen vaihe

Ohjelma siirtyy yhteyden muodostamisen vaiheeseen, kun käyttäjä antaa sellaisen käskyn. Käskyn anto on toteutettu boolean muuttujalla. Tilassa tosi (1) käsky tulkitaan annetuksi ja ohjelma aloittaa yhteyden muodostamisen. Tilassa epätosi (0) yhteys suljetaan ja palataan odottamaan käskyä valmiustilaan. Funktioon on määritettävä IP-osoite ja portti TCP-kirjastoa varten, sekä käyttäjätunnuksen ja salasanan, joilla kirjaudutaan hallintajärjestelmään.

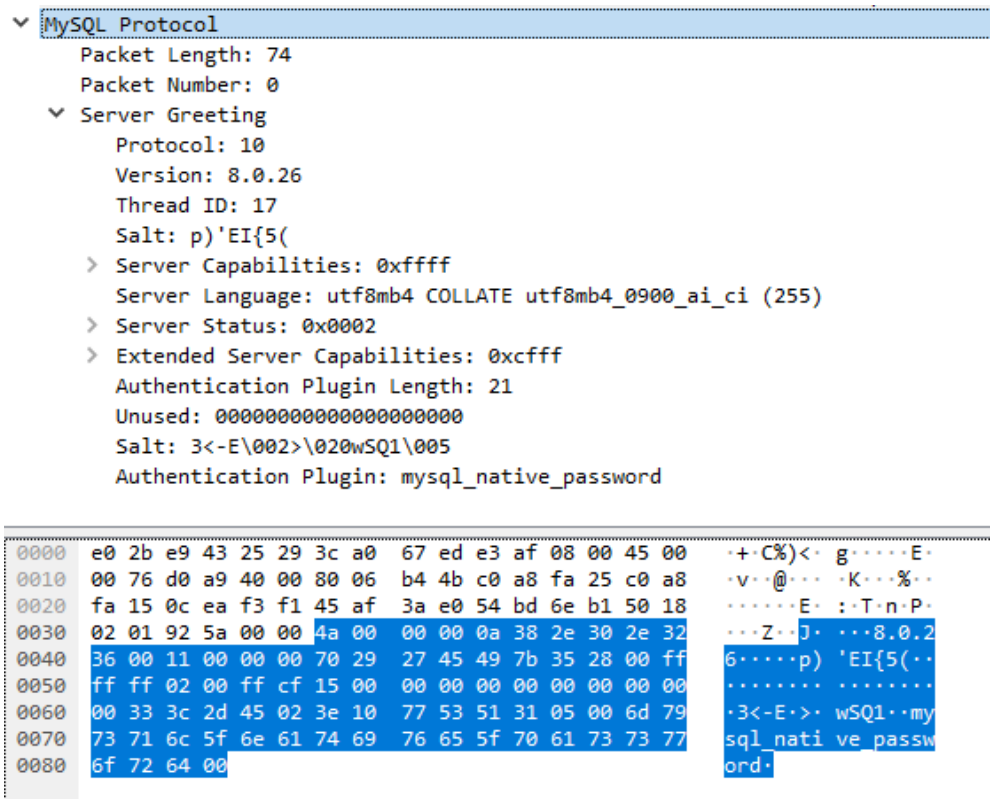
Seuraavana vaiheena on salasanan ensimmäisen osan salaus ja sisäänkirjautumispyyntöpaketin odotus. Sisäänkirjautumispyyntö hallintajärjestelmältä saapuu HandshakeV10-protokollan pakettina, jonka sisältö on esitetty taulukossa 8.

Taulukko 8. HandshakeV10-protokollan paketti (Protocol::HandshakeV10)

Datatyyppi	**Nimi**	**Kuvaus**
int<1>	protocol version	Protokollan versio, aina 10
string<NULL>	server version	MySQL-versio, merkkijonona
int<4>	thread id	Yhteystunnus
string[8]	auth-plugin-data-part-1	Ensimmäinen osa satunnaisdatasta, 8 tavua

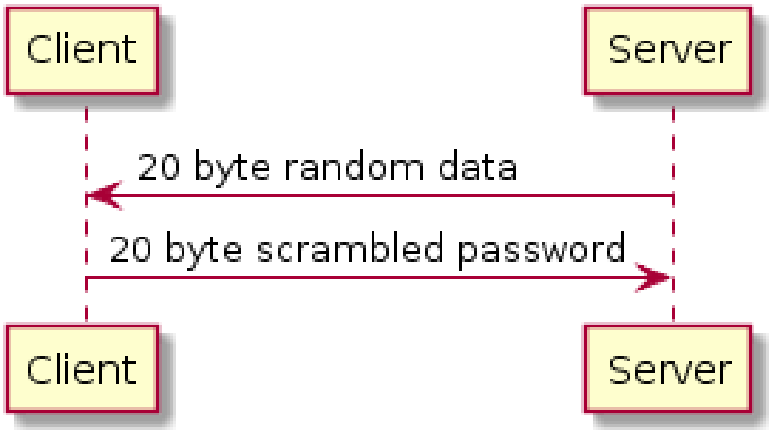
int<1>	filler	0x00 tavu, päättää satunnaisdatan ensimmäisen osan
int<2>	capability_flags_1	Ominaisuusliput 1
int<1>	character_set	Protokollamerkistö
int<2>	status_flags	Tilaliput
int<2>	capability_flags_2	Ominaisuusliput 2
if capabilities & CLIENT_PLUGIN_AUTH {		
int<1>	auth_plugin_data_len	Satunnaisdatan (scramble) kokonaispituus, jos pituus > 0
} else {		
int<1>	00	constant 0x00
}		
string[10]	reserved	Varattu, kaikki tavut 00
$length	auth-plugin-data-part-2	Satunnaisdatan loppuosa,$len=MAX(13, satunnaisdatan kokonaispituus - 8)
if capabilities & CLIENT_PLUGIN_AUTH {		
string<NULL>	auth_plugin_name	Kirjautumismetodin nimi
}		

HandshakeV10-protokollan paketista löytyy tieto protokollasta, palvelimen versiosta, satunnaisdatasta, kirjautumismetodista ja monesta muusta. Kuva 12 esittää palvelimelta saapuvan HandshakeV10-protokollan paketin Wireshark-ohjelmassa.



Kuva 12. Wireshark-ohjelmalla kaapattu HandshakeV10-paketti

Kuvassa 13 esitetty Native Password -sisäänkirjautumismetodi vaatii salasanan kryptauksessa SHA-1 salausalgoritmin sekä palvelimen lähettämän satunnaisdatan käyttöä.



Kuva 13. Native password -sisäänkirjautumismetodi (Native Authentication)

SHA-1 on kryptograafinen tiivistefunktio, joka palauttaa 20-tavuisen hajautusarvon, joka tunnetaan myös nimellä hash. Salausalgoritmia ei rakenneta työssä, sen takia sen toimintaa ei avata tarkemmin. Toimivaksi todettu SHA-1 algoritmi C-kielellä on otettu GitHub-verkkosivustolta. Valmis algoritmi on sovellettu PLC-kirjastoon, johon sisääntulona syötetään STRING-tyyppinen merkkijono ja ulostulona saadaan 20 tavua listamuodossa.

HandshakeV10-protokollan paketissa sijaitseva 20-tavuinen satunnaisdata on jaettu kahdeksi osaksi paketin sisällä. Satunnaisdatan sekä SHA-1 salausalgoritmin avulla käyttäjän salasana salataan kirjautumismetodin määritetyllä tavalla, joka on:

```
SHA1( password ) XOR SHA1( "20-bytes random data from server" <concat>
SHA1( SHA1( password ) ) )
```

(Native Authentication.)

Kirjautumismetodin salausalgoritmi ohjelmassa on pilkottu osiin seuraavalla tavalla:

1. SHA1(salasana)
2. SHA1(Satunnaisdata, SHA1 (SHA1(salasana)))
    a. SHA1 (SHA1(salasana))
    b. Satunnaisdata.

Ohjelmassa kryptattu salasana muodostetaan yllä listatuista osista seuraavassa järjestyksessä:

1. Osa 1 muodostaminen:
    - Syötetään salasana merkkijonona SHA1-algoritmiin, josta saadaan lista 20 tavusta.

2. Osan 2a muodostaminen:

    - Osan 1 muodostuksesta saatu 20 tavua muutetaan merkkijonoksi, joka syötetään SHA-1 algoritmiin toisen kerran. Algoritmista saadaan lista 20 tavusta.

3. Osan 2 muodostaminen:

    - Palvelimelta saatu 20-tavuinen satunnaisdata muutetaan merkkijonoksi, jonka perään liitetään merkkijonoksi muunnettu 2a-osa. Tämä kokonaisuus syötetään SHA-1 algoritmiin, josta saadaan lista 20 tavusta.

4. Kokonaisuuden toteuttaminen:

    - Lopuksi osat 1 ja 2 syötetään XOR-porttiin, josta muodostuu kryptattu salasana, jonka ohjelma lähettää vastauksena palvelimelle.

XOR-portti toimintaa on havainnollistettu totuustauluna taulukossa 9.

Taulukko 9. XOR-totuustaulu

A	**B**	**A XOR B**
0	0	0
0	1	1
1	0	1
1	1	0

XOR on OR-operaattorista eksklusiivinen versio, joka on tosi ainoastaan vain toisen muuttujan (A tai B) ollessa tosi.

Palataan yhteyden muodostusvaiheeseen. Ennen HandshakeV10-protokollan paketin vastaanottoa ohjelma toteuttaa salauksen 1- ja 2a -vaiheet. Tämän jälkeen ohjelma odottaa palvelimen sisäänkirjautumispaketin saapumista.

Satunnaisdata ja kirjautumismetodin nimi ovat ainoat tiedot mitä ohjelma ottaa talteen saapuvasta sisäänkirjautumispaketista. Vaikka kirjautumismetodin nimeä ei hyödynnetä tässä vaiheessa, ohjelman kehittyessä se voi tulla tarpeelliseksi. Esimerkiksi kun tuodaan ominaisuus käyttää erilaisia sisäänkirjautumismetodeja.

Saapuvien TCP-pakettien datan ohjelma vastaanottaa tavulistana. Taulukossa 10 on havainnollistettu miten HandshakeV10-protokollan paketti voisi sijoittua ohjelman tallentamaan listaan.

Taulukko 10. Esimerkki HandshakeV10-paketin paikoista listassa

Tavujen määrä	**Sisältö**	**Paikka listassa**
4	Otsikko	0–3
1	Protokollan versio	4
string[NULL]	Palvelimen versio ('8.0.26' + 00-tavu)	5–11
4	Yhteydentunnus (id)	12–15
string[8]	Satunnaisdatan ensimmäinen osa	16–23
1	Täyte 00-tavu	24
2	Ominaisuusliput	25–26
1	Merkistö	27
2	Tilalippu	28–29
2	Ominaisuusliput	30–31
1	Satunnaisdatan pituus	32
1	00-tavu	33
string[10]	Varattu 00-tavuille	34–43
string[$len]	Satunnaisdatan toinen osa (12 + 00-tavu)	44–56

string[NULL]	Sisäänkirjautumismetodi ('mysql_native_password'+ 00-tavu)	57–78

Paikat listassa vaihtelevat paketin sisällöstä riippuen. Tämän takia ohjelma lukee paketin ottamalla huomioon osat, joiden pituus vaihtelee. MySQL-paketin otsikko koostuu neljästä tavusta, jotka ilmaisivat paketin koon sekä sekvenssinumeron. Otsikon jälkeen ensimmäisenä HandshakeV10-protokollassa on tavu, joka kertoo protokollan version kokonaislukuna. Näiden tavujen ohittaminen on toteutettu määrittämällä ohjelmaan sen, että paketti aloitetaan lukemaan listan viidennestä paikasta. Listassa nolla on myös paikka, joten kuudennes tavu on listan viidennessä paikassa. Protokollan mukaan seuraava tieto on palvelimen versio merkkijonona. Palvelimen versiot voivat vaihdella ja tämä vaikuttaa suoraan merkkijonon pituuteen. Tämän takia emme voi etukäteen tietää merkkijonon pituutta ja datatyyppinä protokollassa on string<NULL>. Palvelimen versio luetaan while-silmukalla, joka siirtyy listassa eteenpäin paikka kerralla, kunnes luettu tavu on nolla. Nollatavu string<NULL>-datatyypissä tarkoitti merkkijonon päättymistä. Palvelimen version jälkeen seuraava on nelitavuinen yhteystunnus. Ohjelma on jäänyt while-silmukassa nollatavuun, siksi siirrytään listassa viisi paikkaa eteenpäin, jolloin päästään satunnaisdatan ensimmäiseen osaan. Ensimmäinen osa koostuu aina kahdeksasta tavusta, jotka tallennetaan for-silmukalla erilliseen listaan. Tiedoilla satunnaisdatan ensimmäisen ja toisen osan välissä on kiinteä pituus. Siirtymällä listassa eteenpäin näiden tietojen kokonaispituuden verran, ohjelma pääsee viimeisen satunnaisdatan osaan, jonka pituus on 12 tavua. Pituus on saatu vähentämällä ensimmäisen osan pituuden (8 tavua) satunnaisdatan kokonaispituudesta (20 tavua). Vaikka HandshakeV10-protokollan paketissa satunnaisdatan kokonaispituudeksi on ilmoitettu 21 tavua, viimeisenä tavuna on 00-tavu, joka on jätettävä huomioimatta. Nollatavun jälkeen on kirjautumismetodin nimi merkkijonona, jonka ohjelma ottaa talteen kirjaston mahdollista kehitystä varten. Sisäänkirjautumispaketin lukeminen päättyy tähän.

Seuraavaksi ohjelma siirtyy muodostamaan kryptatun salasanan viimeisen osan. Ohjelma liittää kahteen kertaan SHA-1 algoritmissa (osa 2a, SHA1(SHA1 (salasana))) salatun salasanan satunnaisdatan perään. Tämä kokonaisuus

muutetaan merkkijonoksi ja salataan SHA-1-algoritmilla. Tässä vaiheessa ohjelma pitää muistissa salauksen molemmat osat. Viimeisenä vaiheena ohjelma syöttää osat 1 ja 2 XOR-porttiin. Tämän suoritettua ohjelmalla on 20-tavuinen data, joka vastaa toteutettua kirjautumismetodin salausalgoritmia. Juuri tämän tavulistan ohjelma lähettää vastauspaketin yhteydessä palvelimelle.

Vastauspaketin, jonka ohjelma seuraavaksi rakentaa on oltava HandshakeResponse41-protokollan mukainen. Tämän protokollan sisältö on esitetty taulukossa 11.

Taulukko 11. HandshakeResponse41-protokollan paketti (Protocol::HandshakeResponse)

Datatyyppi	**Nimi**	**Kuvaus**
int<4>	client_flag	Ominaisuusliput, CLIENT_PROTOCOL_41 lipun aina käytettävä
int<4>	max_packet_size	Paketin suurin koko
int<1>	character_set	Käyttäjän käyttämä merkistö
string[23]	filler	Täyte, kaikki tavut 00
string<NUL>	username	Käyttäjätunnus
if capabilities & CLIENT_PLUGIN_AUTH_LENENC_CLIENT_DATA {		
string<length>	auth_response	Salattu salasana, joka on luotu kirjautumismetodin osoittamalla tavalla
} else {		
int<1>	auth_response_length	Salatun salasanan pituus
string<length>	auth_response	Salattu salasana, joka on luotu kirjautumismetodin osoittamalla tavalla
}		
if capabilities & CLIENT_CONNECT_WITH_DB {		

string<NUL>	database	Tietokanta, joka valitaan sisäänkirjautumisessa
}		
if capabilities & CLIENT_PLUGIN_AUTH {		
string<NUL>	client_plugin_name	Käytetty sisäänkirjautumismetodi vastauspaketissa
}		
if capabilities & CLIENT_CONNECT_ATTRS {		
int<lenenc>	length of all key-values	Kaikkien asiakasattribuuttien arvojen pituus
string<lenenc>	key1	Ensimmäisen asiakasmääritteen nimi
string<lenenc>	value1	Ensimmäisen asiakasattribuutin arvo
.. (if more data in length of all key-values, more keys and values parts)		
}		

Ohjelma muodostaa vastauspaketin listaan, jossa yksi paikka vastaa yhtä tavua. Paketin muodostaminen alkaa otsikosta. Paketin pituus määritetään lopuksi ja sille varataan kolme ensimmäistä paikkaa listassa. Neljäs paikka listassa on varattu sekvenssinumerolle. Sen arvo on 1, koska ohjelma lähettää vastauksen vastaanotetulle paketille eli samaan asiaan kuuluva kommunikointi. Seuraavaksi ohjelma määrittää seuraavat ominaisuusliput:

- CLIENT_PROTOCOL_41
- CLIENT_SECURE_CONNECTION
- CLIENT_TRANSACTIONS.

Ensimmäinen ominaisuuslippu ilmoittaa palvelimelle, että PLC-kirjasto tukee HandshakeResponse41-protokollan. Toisin sanoen sen mukainen paketti lähetetään palvelimeen. Toinen ominaisuuslippu ilmoittaa palvelimelle, että Native Password -sisäänkirjautumismetodi tuetaan. Viimeinen ominaisuuslippu on pakollinen ja sallii lähettämään palvelimelle tilalippuja EOF-paketissa. Ohjelma lisää CLIENT_CONNECT_WITH_DB-ominaisuuslipun, jos käyttäjä määrittää tietokannan, johon yhdistetään yhteyden muodostaessa. Ominaisuuslipuilla on oma arvonsa, joka on määritelty dokumentaatiossa. Näitä arvoja eli ominaisuuslippuja ohjelma yhdistää yhdeksi arvoksi vastauspakettia varten käyttäen OR-operaattoria. Dokumentaatiossa ominaisuuslippujen arvot ovat esitetty heksadesimaalilukuina, mutta ohjelmaan ne on syötetty kokonaislukuina, jonka ohjelma muuttaa tavuiksi. Toteutus ohjelmassa näyttää tältä:

```
IF Database <> '' THEN
     cap_flag := 512 OR 8192 OR 32768 OR 8;
ELSE
     cap_flag := 512 OR 8192 OR 32768;
END_IF
memcpy(ADR(ResponsePacketSend[4]), ADR(cap_flag), SIZEOF(cap_flag));
```

Seuraava tieto, joka lisätään listaan ilmoittaa paketin suurimman koon. Arvo annetaan tavuna ja oletuksena se on 1 GB eli 1 073 741 824 tavua. Tiedot ominaisuuslipuista sekä paketin koosta varaavat neljä tavua kukin. Seuraavana tietona listassa on merkistö, joka on asetettu olevan latin1_swedish_ci. Tämä merkistö on valittu sen takia, koska MySQL (versio 5.7) käyttää sitä oletuksena tietokannoissa. Seuraavat 23 tavua on täytettä, joka koostuu 00-tavuista. Täytteen jälkeen ohjelma kirjaa listaan käyttäjätunnuksen, joka päätetään 00-tavulla, koska datatyyppinä on string<NULL>. Käyttäjätunnuksen perään ohjelma kirjaa kryptatun salasanan pituuden, joka on 20 tavua. Listan seuraavaan paikkaan ohjelma kopioi aiemmin muodostetun kirjautumismetodin mukaan kryptatun salasanan. Jos käyttäjä on määrittänyt tietokannan, niin ohjelma kirjaa sen salasanan jälkeen päättäen 00-tavullaa. Lopuksi ohjelma laskee muodostetun paketin pituuden ja merkitsee sen otsikkoon. Muuta vastauspakettiin ei tule, koska muita ominaisuuksia PLC-kirjasto ei tue. Koottu lista lähetetään palvelimelle, jonka jälkeen siirrytään odottamaan sen vastauksen.

Vastauksena hallintajärjestelmä lähettää joko OK- tai ERR-paketin. Ohjelma ohittaa otsikon ja siirtyy listan neljänteen paikkaan, joka on dokumentaation mukaan joko 00- tai ff-tavu. Tavun ollessa 00, ohjelma toteaa, että sisäänkirjautuminen on onnistunut, jonka jälkeen se siirtyy komentovaiheeseen. Tavun ollessa ff, ohjelma lukee paketin virheen ja virhekoodin, jonka jälkeen palauttaa virhetiedot käyttäjälle ja sulkee yhteyden.

Yhteyden muodostamisen vaihe oli ehdottomasti opinnäytetyön haastavin osa. SHA-1 algoritmin liittäminen PLC-ohjelmaan sekä sen soveltaminen kirjautumismetodin mukaan oli haasteista ensimmäinen. Käyttääkseen kirjaston funktion muuttujia (VAR_INPUT, VAR_OUTPUT) C-ohjelmassa, B&R:n Automation Studiossa muuttujan eteen on laitettava inst-> -merkkijono. Sisäänkirjautumismetodissa on osa, jossa salasana syötetään SHA-1 algoritmiin kahteen kertaan. Algoritmista saatu lista 20 tavusta on muutettava string-tyyppiseen merkkijonoon. Ohjelmassa muutos on toteutettu kopioimalla listan muistialueen string-tyyppiseen muuttujaan seuraavalla tavalla:

```
memcpy(ADR(SHA1Input), ADR(stage2_hash), SIZEOF(stage2_hash));
```

Palvelin säilyttää käyttäjien salasanat SHA1(SHA1(salasana)) -muodossa, joka saadaan näkyville hallintajärjestelmässä suorittamalla SELECT * FROM mysql.user; -komentoa. Vertailemalla palvelimessa säilyvän salasanan ohjelman toteutetun salausosan kanssa voidaan todeta, onko kryptaus tehty oikein vai väärin.

Toisena haasteena oli vastauspaketin lähetys. Käytetty TCP-kirjasto on lähettänyt ensin pelkkiä nollia sisältämän vastauspaketin, jonka jälkeen lähetti ohjelman muodostetun vastauspaketin. Palvelin on vastaanottanut tyhjän paketin, josta se ilmoitti virheen sekä lähetti uuden sisäänkirjautumispyyntöpaketin uudella satunnaisdatalla. Vasta tämän jälkeen paketti vanhentuneella satunnaisdatalla kryptatulla salasanalla on saapunut palvelimelle. Tämän seurauksena palvelin on lähettänyt virheen väärästä salasanasta. Ongelma on huomattu analysoimalla päätepisteiden kommunikointia Wireshark-ohjelmassa. TCP-kirjastoa on paljon korjattu ja kehitetty opinnäytetyön aikana.

6.2.2 Komentovaihe

Onnistuneen sisäänkirjautumisen jälkeen ohjelma siirtyy komentovaiheeseen. Komentovaiheessa ohjelma lähettää SQL-kyselyitä hallintajärjestelmään sekä käsittelee tulevia vastauspaketteja. Ohjelma on valmiustilassa, kunnes käyttäjä ilmoittaa haluavansa katkaista yhteys palvelimeen tai lähettää SQL-kyselyn palvelimelle. Komennon lähettäminen vaatii käyttäjältä SQL-syntaksin mukaisen kyselyn merkkijonona. Ohjelma rakentaa Text-protokollaan kuuluvan COM_QUERY-komennon, joka on esitetty taulukossa 12. Ohjelma ilmoittaa käyttäjälle olevansa varattu, kun ohjelma ei ole valmiustilassa. Tilatiedon avulla käyttäjä tietää voiko lähettää uuden SQL-kyselyn tai onko palvelimen vastauspaketti luettu loppuun.

Taulukko 12. COM_QUERY-paketti (COM_QUERY)

Datatyyppi	**Nimi**	**Kuvaus**
int<1>	Command	Komento 0x03
string<EOF>	Query	Suoritettava SQL-kyselyn teksti

Ohjelma rakentaa tavu kerralla listaan myös komentovaiheessa. Ensimmäisenä on paketin koosta ja sekvenssinumerosta koostuva otsikko. Sekvenssinumero alkaa aina nollasta, koska jokainen komento on itsenäinen kokonaisuus eikä se liity edelliseen kommunikointiin. Seuraavaksi ohjelma lisää listaan kokonaisluvun 3, joka vastaa COM_QUERY-komentoa Text-protokollassa. Kokonaisluvun perään ohjelma liittää käyttäjän syöttämän SQL-kyselyn merkkijonona. Ohjelma laskee paketin pituuden ja liittää sen otsikkoon, jonka jälkeen lähettää muodostetun paketin palvelimelle.

Vastauksena kyselykomentoon voi olla jokin seuraavista paketeista:

- ERR-paketti
- OK-paketti

- LOCAL INFILE pyyntö (opinnäytetyön aiheen ulkopuolella)

- Text Resultset vastaus.

OK- ja ERR-paketteja sekä niiden lukemista on käsitelty luvussa 5.1. Tässäkin tapauksessa lukumetodit ovat samanlaiset eli ohjelma lukee tavut listasta. Näiden pakettien lukemisen jälkeen ohjelma ilmoittaa käyttäjälle joko onnistuneesta komennon suorittamisesta tai tapahtuneesta virheestä, jonka jälkeen palaa komennon suorittamisen valmiustilaan. Virhetilanteessa ohjelma ilmoittaa käyttäjälle virhetekstin sekä virhekoodin. Palvelin palauttaa OK-paketin, jos komento on liittynyt tietojen muokkaamiseen, lisäämiseen tai poistamiseen.

Palvelin lähettää Text Resultset -vastauksen, jos SQL-kysely liittyy tiedonhakuun, jonka on välitettävä käyttäjälle (esimerkiksi SELECT * FROM taulukko;). Ohjelma olettaa, että kyseessä on Text Resultset -vastaus, jos saapuva paketti ei tunnisteta OK- tai ERR-paketiksi. Text Resultset koostuu monista paketeista, jonka ovat:

- sarakkeiden määrän ilmoittava paketti

- sarakkeen tietoja sisältävä paketti, jokaisen sarakkeen tiedot saapuvat omana pakettina

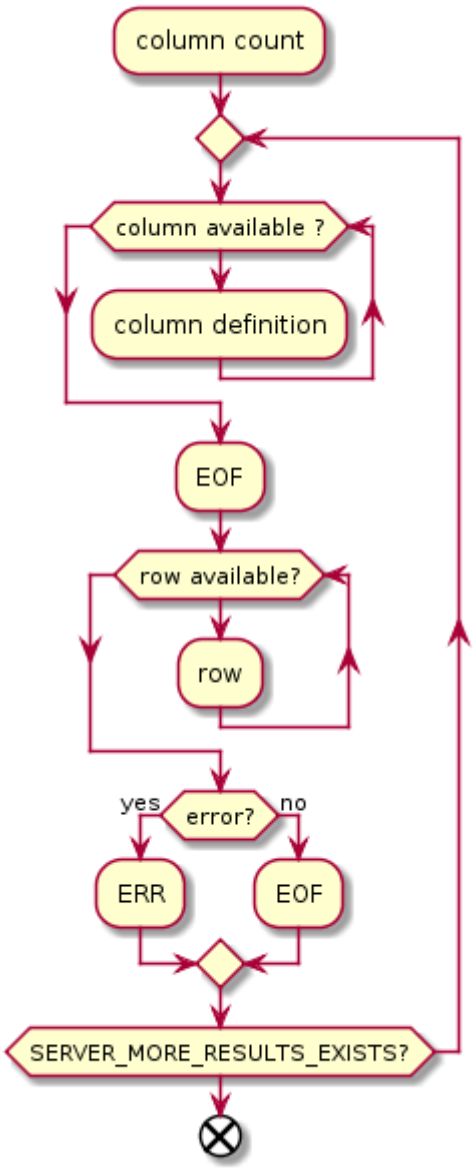
- EOF-paketti

- rivintieto paketti, joka sisältää arvot jokaisesta sarakkeesta, jokainen rivi paketti saapuu omana pakettina

- ERR- tai EOF-paketti.

Text Resultset -vastauksen tapauksessa vastaanotettu TCP-paketti sisältää useita MySQL-palvelimen paketteja. Toisin sanoen ohjelma jatkaa uusien MySQL-pakettien lukemisen samasta listasta, joita TCP-paketti sisältää. Uusi

TCP-paketti vastaanotetaan, kun ohjelma on päässyt listan viimeiseen paikkaan. Kuvassa 14 on esitetty, miten palvelin muodostaa Text Resultset -vastauksen, jonka voi myös pitää lukuohjeena.



Kuva 14. Text Resultset -vastauksen sisältö (Text Resultset)

Ohjelman Text Resultset -vastauksen lukeminen alkaa lyhyestä paketista, joka koostuu ainoastaan otsikosta (paketin pituus ja sekvenssinumero) ja int<lenenc>-tyyppisestä kokonaisluvusta, joka ilmaisee vastauksessa olevien sarakkeiden määrän. Seuraavana listassa on ColumnDefinition41-protokollan paketti,

joka sisältää tiedot sarakkeista. Protokollan paketin tarkka sisältö on kuvattu taulukossa 13.

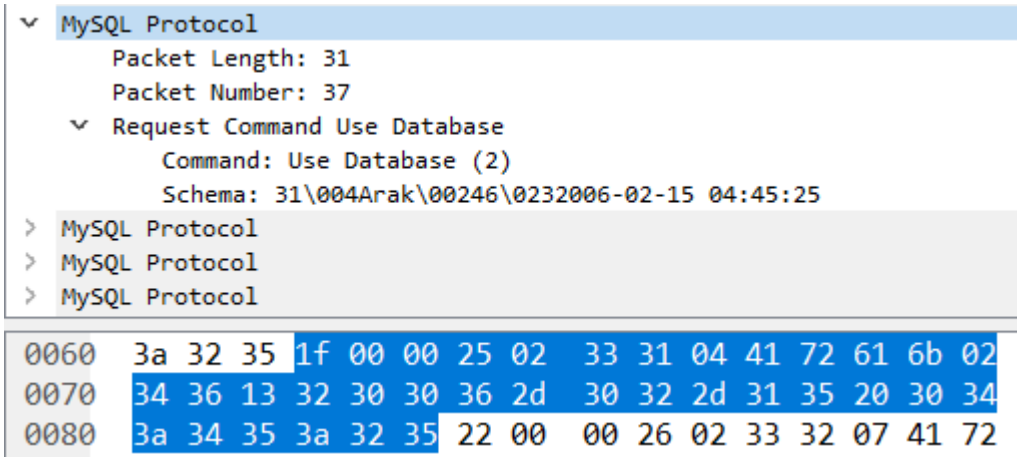
Taulukko 13. ColumnDefinition41-protokolla (Column Definition)

Datatyyppi	**Nimi**	**Kuvaus**
string<lenenc>	catalog	aina "def"
string<lenenc>	schema	tietokannan nimi
string<lenenc>	table	taulukon virtuaalinen nimi
string<lenenc>	org_table	taulukon fyysinen nimi
string<lenenc>	name	sarakkeen virtuaalinen nimi
string<lenenc>	org_name	sarakkeen fyysinen nimi
int<lenenc>	length of fixed length fields	[0x0c]
int<2>	character_set	sarakkeen käyttämä merkistö
int<4>	column_length	sarakkeen kentän enimmäispituus
int<1>	type	sarakkeen datatyyppi
int<2>	flags	Ominaisuusliput
int<1>	decimals	Näytetyt desimaaliluvut: 0x00 kokonaisluvuille 0x1f dynaamisille merkkijonoille 0x00 - 0x51 desimaaleille

Tässä vaiheessa ohjelma pitää muistissa sarakkeiden määrän, joka myös kuvaa ColumnDefinition41-protokollan pakettien määrän, joita luetaan. Ohjelma on rakennettu lukemaan sarakkeiden tiedot for-silmukassa, joka päättyy, kun ohjelma on lukenut paketteja sarakkeiden verran. Ohjelma kulkee paketin läpi hyödyntämällä kiinteäpituisia tietoja sekä lukemalla string<lenenc>-arvoja, jotka tarkoittavat seuraavan tiedon pituutta. Jokaisella silmukan kierroksella ohjelma

tallentaa erilliseen listaan taulukon nimen, sarakkeen nimen ja sarakkeen datatyypin. Ohjelma välittää tämän listan käyttäjälle. Sen avulla käyttäjä osaa tulkita toisen listan, johon ohjelma kerää rivitiedot. Saraketietojen luettua, ohjelma ohittaa EOF-paketin ja siirtyy seuraavaan vaiheeseen, jossa luetaan rivitietoja.

Rivitiedot sisältävät jokaisen sarakkeen tiedot. Rivitietopaketti koostuu otsikosta ja sarakkeiden arvoista. Sarakkeiden arvot ovat paketissa samassa järjestyksessä missä ColumnDefinition41-protokollan paketit ovat olleet. Kaikki arvot paketissa ovat string<lenenc>-tyyppisiä, poikkeuksena on NULL-arvo, joka ilmaistaan fb-tavuna. Käytännössä tämä tarkoittaa, että merkkijonon edessä on kokonaisluku, joka ilmoittaa merkkijonon pituuden. Ohjelma tallentaa rivitiedot kaksiulotteiseen listaan, josta ensimmäinen arvo vastaa saraketta ja toinen arvo vastaa sarakkeen rivitiedon arvoa. Kuvassa 15 on Wireshark-ohjelmalla kaapattu rivitietopaketti.



Kuva 15. Esimerkki rivitietopaketista

Ohjelma lukee kuvan esimerkkipaketin (korostettu sinisellä) seuraavalla tavalla (ohjelma tietää tässä vaiheessa sarakkeiden lukumäärän):

- Ohitetaan otsikko-osa (1f 00 00 25), siirrytään listassa 4 paikkaa eteenpäin.

- Ensimmäisen sarakkeen rivitiedon pituus on 2 tavua (tavu 02 muunnettuna luvuksi).

- Ensimmäisen sarakkeen rivitieto on ’31’ (tavu 33 ja 31 muutettuna merkkijonoon).

- Toisen sarakkeen rivitiedon pituus on 4 tavua (tavu 04 muunnettuna luvuksi).

- Toisen sarakkeen rivitieto on ’Arak’ (tavut 41, 72, 61 ja 6b muunnettuna merkkijonoksi).

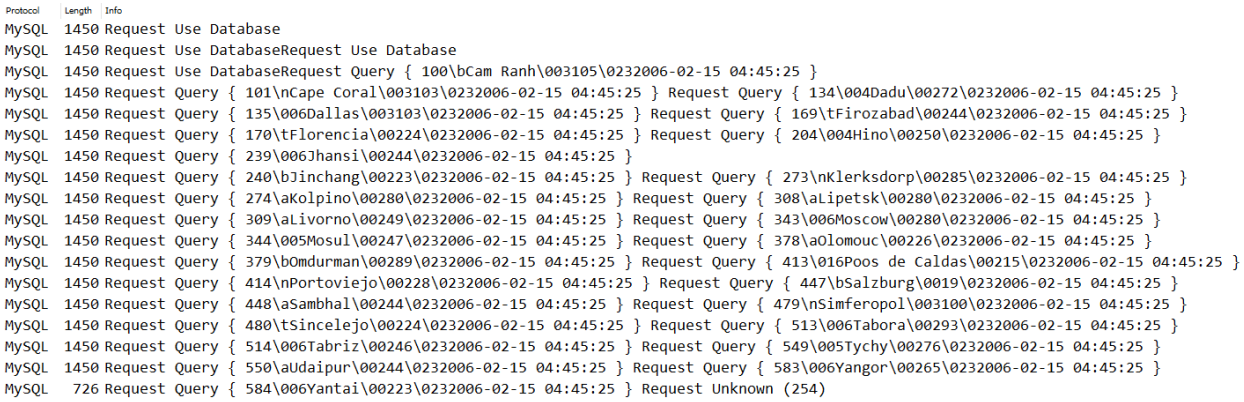
- Kolmannen sarakkeen rivitiedon pituus on 2 tavua (tavu 02 muunnettuna luvuksi).

- Kolmannen sarakkeen rivitieto on ’46’ (tavut 34 ja 36 muunnettuna merkkijonoksi).

- Viimeisen sarakkeen rivitiedon pituus on 23 tavua (tavut 13 muunnettuna merkkijonoksi).

- Viimeisen sarakkeen rivitieto on ’2006-02-15 04:45:25’ (paketin loput tavut muunnettuna merkkijonoksi).

Ohjelma lukee rivitietopaketin for-silmukassa sarakkeiden määrään verran, jonka jälkeen siirtyy seuraavaan rivitietopakettiin. MySQL-palvelimelta rivitietoja voi tulla niin paljon, että niitä lähetetään useampana TCP-pakettina, kuten kuvassa 16 näkyy.



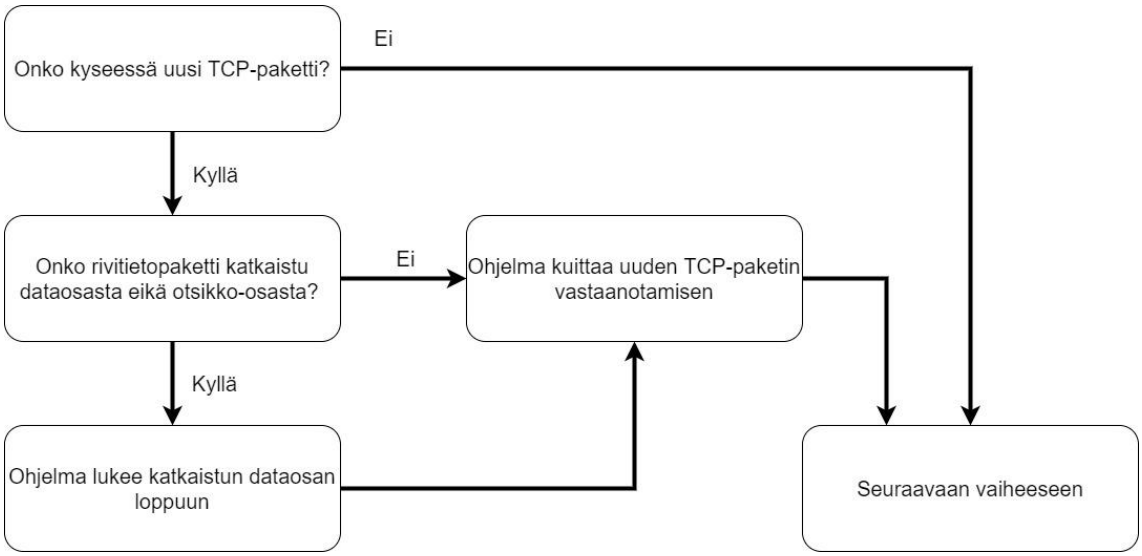
Protocol	Length	Info
MySQL	1450	Request Use Database
MySQL	1450	Request Use DatabaseRequest Use Database
MySQL	1450	Request Use DatabaseRequest Query { 100\bCam Ranh\003105\0232006-02-15 04:45:25 }
MySQL	1450	Request Query { 101\nCape Coral\003103\0232006-02-15 04:45:25 } Request Query { 134\004Dadu\00272\0232006-02-15 04:45:25 }
MySQL	1450	Request Query { 135\006Dallas\003103\0232006-02-15 04:45:25 } Request Query { 169\tFirozabad\00244\0232006-02-15 04:45:25 }
MySQL	1450	Request Query { 170\tFlorencia\00224\0232006-02-15 04:45:25 } Request Query { 204\004Hino\00250\0232006-02-15 04:45:25 }
MySQL	1450	Request Query { 239\006Jhansi\00244\0232006-02-15 04:45:25 }
MySQL	1450	Request Query { 240\bJinchang\00223\0232006-02-15 04:45:25 } Request Query { 273\nKlerksdorp\00285\0232006-02-15 04:45:25 }
MySQL	1450	Request Query { 274\aKolpino\00280\0232006-02-15 04:45:25 } Request Query { 308\aLipetsk\00280\0232006-02-15 04:45:25 }
MySQL	1450	Request Query { 309\aLivorno\00249\0232006-02-15 04:45:25 } Request Query { 343\006Moscow\00280\0232006-02-15 04:45:25 }
MySQL	1450	Request Query { 344\005Mosul\00247\0232006-02-15 04:45:25 } Request Query { 378\aOlomouc\00226\0232006-02-15 04:45:25 }
MySQL	1450	Request Query { 379\bOmdurman\00289\0232006-02-15 04:45:25 } Request Query { 413\016Poos de Caldas\00215\0232006-02-15 04:45:25 }
MySQL	1450	Request Query { 414\nPortoviejo\00228\0232006-02-15 04:45:25 } Request Query { 447\bSalzburg\0019\0232006-02-15 04:45:25 }
MySQL	1450	Request Query { 448\aSambhal\00244\0232006-02-15 04:45:25 } Request Query { 479\nSimferopol\003100\0232006-02-15 04:45:25 }
MySQL	1450	Request Query { 480\tSincelejo\00224\0232006-02-15 04:45:25 } Request Query { 513\006Tabora\00293\0232006-02-15 04:45:25 }
MySQL	1450	Request Query { 514\006Tabriz\00246\0232006-02-15 04:45:25 } Request Query { 549\005Tychy\00276\0232006-02-15 04:45:25 }
MySQL	1450	Request Query { 550\aUdaipur\00244\0232006-02-15 04:45:25 } Request Query { 583\006Yangor\00265\0232006-02-15 04:45:25 }
MySQL	726	Request Query { 584\006Yantai\00223\0232006-02-15 04:45:25 } Request Unknown (254)

Kuva 16. MySQL-palvelimen lähettämä Text Resultset vastaus

Kuvasta 16 huomataan, että MySQL-palvelimen lähettämät TCP-paketit ovat 1450 tavun pituisia ja ainoastaan viimeinen paketti on pienempi. MySQL-palvelin siis käyttää kaiken TCP-paketissa olevan tilan hyödyksi. Text Resultset -vastaukselle tämä tarkoittaa sen, että mikä tahansa paketti voi katkea mistä tahansa kohdasta. Katkaistu paketti jatkuu seuraavassa TCP-paketissa katkeamiskohdasta. Rivitietopaketti voi myös päätyä TCP-paketin viimeiseen paikkaan, jolloin katkeamista ei tapahdu ja ohjelma vastaanottaa uuden TCP-paketin. Text Resultset -vastauksen päättävä EOF-paketti voi myös olla jaettuna kahteen TCP-pakettiin. Kaikki mahdolliset vaihtoehdot on huomioitu ohjelmassa.

Ohjelma pitää muistissa vaiheen mihin se on jäänyt saavuttua TCP-paketin listan viimeiselle paikalle. Tämän ansiosta vastaanottamalla seuraavan TCP-paketin ohjelma osaa jatkaa datan tallentamista oikeaan paikkaan.

Ohjelmassa on kolme vaihetta, joiden avulla se huomioi kaikki tilanteet. Ensimmäinen osa on toteutettu kuvassa 17 olevan vuokaavion mukaan. Sen avulla ohjelma kuittaa vastaanotetun uuden TCP-paketin sekä lukee katkaistut rivitietopaketin saraketiedon loppuun, jos sille on tarvetta.



Kuva 17. Text Resultset -vastauksen lukemisen ensimmäinen vaihe

Toisessa vaiheessa ohjelma lukee TCP-paketin sisältämät rivitietopaketit ja tallentaa tiedot listaan. Tässä vaiheessa ohjelma myös käsittelee paketin katkaisun. Aluksi ohjelma tarkastelee, onko seuraavana pakettina EOF-paketti, joka tarkoitti Text Resultset -vastauksen päättymisen. Näin ollessa ohjelma siirtyy komentovaiheen valmiustilaan ja ilmoittaa käyttäjälle kyselyn loppuun suorituksesta. Ohjelma jatkaa eteenpäin, jos seuraavana on rivitietopaketti. Tämän jälkeen ohjelma tarkastelee missä sarakkeiden laskurissa mennään. Laskurin arvo alle sarakkeiden määrän tässä vaiheessa tarkoittaisi, että jatketaan edellisen TCP-paketin rivitietopaketin lukemista. Laskurin arvon ollessa yli sarakkeiden määrän tarkoittaa sen, että edellinen rivitietopaketti on luettu loppuun. Tällöin ohjelma nollaa laskurin ja siirtyy lukemaan seuraavan paketin, joka alkaa otsikon tarkastelusta. Ohjelma siirtyy vastaanottamaan uuden TCP-paketin, jos TCP-paketin lista katkeaa otsikosta. Muussa tapauksessa otsikko ohitetaan. Seuraavaksi ohjelma menee for-silmukkaan, jossa luetaan rivitietopaketin tiedot. Jokaisen sarakkeen tiedon keräämisen yhteydessä ohjelma tarkistaa onko tavuna fb eli NULL-arvo tai katkeaako TCP-paketti kerättävään rivitietoon.

Kolmas vaihe on toteutettu omassa CASE-valintarakenteessa, jossa ohjelma pyytää TCP-kirjastolta vastaanottamaan seuraavan paketin. TCP-kirjasto ilmoit-

taa vastaanottavan uuden paketin, jolloin ohjelma siirtyy ensimmäiseen vaiheeseen. Näiden vaiheiden sykli jatkuu niin kauan, kunnes ohjelma pääsee Text Resultset -vastauksen päättävän EOF-pakettiin, jonka luettua ohjelma siirtyy komentovaiheen.

Komentovaiheen toteutus on ollut haastava. Monien poikkeustilanteiden huomiointi on onnistunut laajan testauksen avulla. Text Resultset -vastauksen lukemisessa oli eniten haasteita. Aluksi tilanteessa, jossa MySQL-hallintajärjestelmä lähetti samanaikaisesti useita TCP-paketteja, TCP-kirjasto vastaanotti niitä hallittomasti, jolloin yksittäiset paketit saattoivat jäädä lukematta. TCP-kirjasto on kehitetty luomalla siihen ominaisuuden, joka mahdollistaa pakettien hallitun vastaanoton.

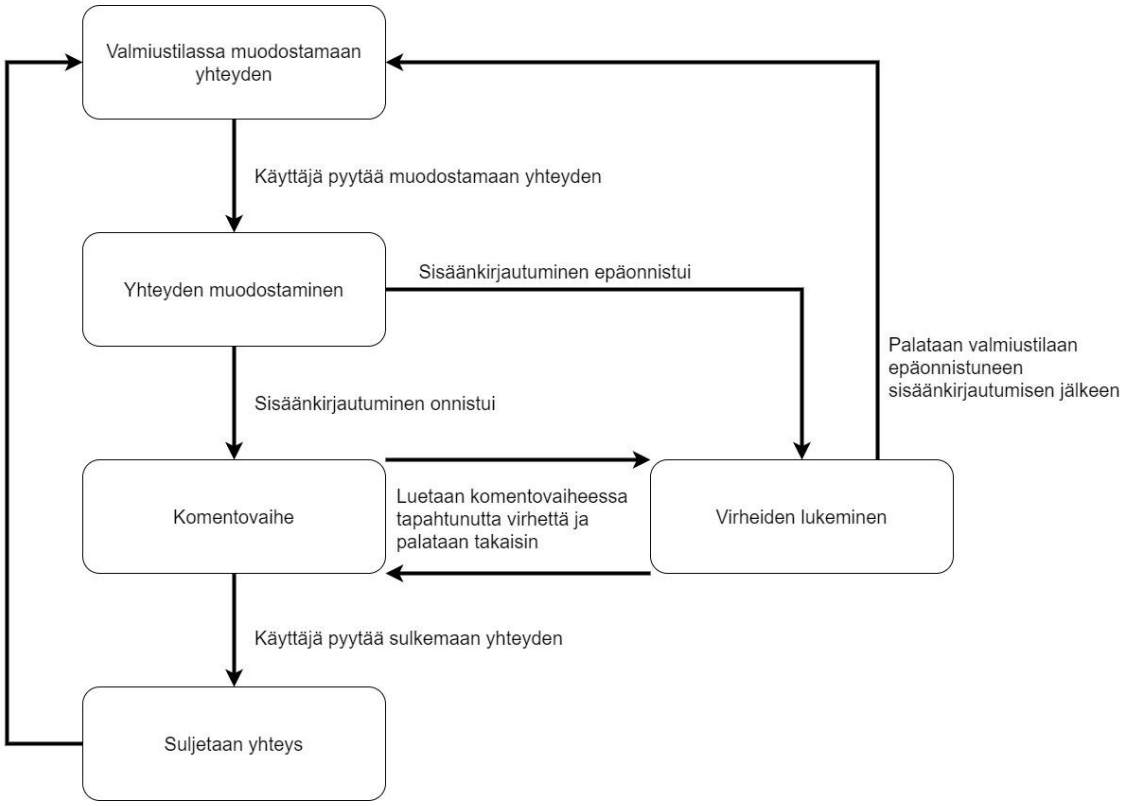
6.2.3 Yhteyden sulkeminen

Yhteyden sulkemisen vaiheessa ohjelma lähettää COM_QUIT-komennon ja alustaa useita muuttujia uuden yhteyden luomisen mahdollistamiseksi. COM_QUIT-komennon avulla kerrotaan palvelimelle, että halutaan sulkea yhteys. Komento koostuu otsikosta ja int<1>-tyyppisestä tavusta, jonka arvo on 1. Tämän jälkeen ohjelma siirtyy yhteyden luomisen CASE-valintarakenteessa olevaan valmiustilaan.

6.3 Microsoft SQL Server -funktion suunnittelu

Kirjastossa on käytetty B&R:n valmista AsDb-kirjastoa, joka toteuttaa kommunikoinnin Microsoft SQL Server -hallintajärjestelmän kanssa. Funktion tarkoituksena on yksinkertaistaa valmiin PLC-kirjaston. Yksinkertaistus toteutetaan konfiguroimalla ja suorittamalla valmiin kirjaston funktioita ohjelmallisesti käyttäjän sijasta. Koko toiminnallisuus on suunniteltu toteutumaan yhdellä funktiolla.

Microsoft SQL Server -funktion suunnittelussa on sovellettu MySQL-funktion rakennetta. Funktio on toteutettu kuvassa 18 esitetyn vuokaavion mukaan.



Kuva 18. Microsoft SQL Server -funktion vuokaavio

Ohjelma alkaa yhteyden muodostamisen vaiheesta. Tämäkin vaihe alkaa valmiustilasta, jossa ohjelma odottaa käyttäjän pyyntöä muodostaa yhteyden hallintajärjestelmään. Yhteyden muodostamisen jälkeen ohjelma siirtyy komentovaiheeseen, johon se jää, kunnes käyttäjä pyytää sulkemaan yhteyden. Komentovaiheessa ohjelma suorittaa käyttäjän antamia SQL-kyselyitä sekä välittää palvelimelta saapuvia vastauksia. Käyttäjän pyynnöstä ohjelma voi sulkea yhteyden hallintajärjestelmään, jonka jälkeen se palaa valmiustilaan muodostamaan uuden. Tapahtuneita virheitä ohjelma lukee omassa CASE-valintarakenteessa, josta se palaa takaisin joko komentovaiheeseen tai yhteyden muodostamisen valmiustilaan, riippuen siitä missä virhe on tapahtunut.

6.4 Microsoft SQL Server -funktion toteutus

Käyttäjän pyynnöstä ohjelma siirtyy valmiustilasta muodostamaan yhteyden dbConnect-funktion avulla. Ohjelma seuraa funktioiden statukset (esimerkiksi

dbConnect.status), josta se saa tiedon funktion tilasta ja siirtyy seuraavaan vaiheeseen arvon perusteella. Statuksen arvot, joita ohjelmassa otetaan huomioon, on esitetty taulukossa 14.

Taulukko 14. Funktioiden statuksen arvot

Statuksen arvo	**Selite**
0	Virhettä ei ole, funktio on suoritettu loppuun
65535	Varattu, funktion suorittaminen on vielä kesken
-1	Yleinen virhe. Tarkemman tiedot saatavilla virhefunktiosta
100	Dataa ei ole saatavilla

Muut statuksen arvot ohjelma luokittelee virheiksi ja siirtyy lukemaan niitä virheitä käsittelevään CASE-valintarakenteeseen.

Yhteyden muodostettua ohjelma siirtyy komentovaiheen valmiustilaan, jossa se odottaa käyttäjän SQL-kyselyä ja pyyntöä suorittaa sen suorittamiseen. SQL-kyselyn ohjelma suorittaa dbExecute-funktiolla, jonka kutsuttua ohjelma siirtyy seuraamaan funktion statusta.

Ohjelma lukee palvelimen vastauksen dbGetAffectedRows-funktiolla, jos SQL-kysely ei liittynyt tiedonhakuun. Tällä funktiolla kerrotaan käyttäjälle, kuinka moneen riviin kysely on vaikuttanut.

Palvelimen vastaus luetaan kahdella funktiolla, jos SQL-kysely liittyi tiedonhakuun. Nämä funktiot ovat dbGetData ja dbFetchNextRow. Ensimmäisestä funktiosta ohjelma saa rivitiedot sarake kerrallaan. Kaikkien sarakkeiden tiedot luettua toisen funktion avulla ohjelma siirtyy seuraavaan riviin. Ohjelma suorittaa näitä funktiota, kunnes dbFetchNextRow-funktion statuksesta saadaan virhearvon 100, joka tarkoittaa kaiken datan olevan luettu. Palvelimelta tuleva data tallennetaan kaksiulotteiseen listaan, josta ensimmäinen indeksi on sarake ja toinen on sarakkeen rivitieto.

Virheitä luetaan dbGetErrorMessage-funktiolla, josta ohjelma välittää käyttäjälle sekä virhekoodin että viestin. Yhteyden sulkemisen ohjelma toteuttaa dbDisconnect-funktiolla, jonka jälkeen palaa valmiuteen muodostamaan uuden yhteyden.

6.5 Kirjaston kehitysideoita

PLC-kirjaston toteutuksen aikana sekä sen jälkeen on syntynyt useita kehitysideoita. Aiheen rajauksen ja aikataulun takia kaikkia kehitysideoita ei kuitenkaan voitu toteuttaa.

MySQL-funktioon olisi hyvä lisätä mahdollisuuden käyttää muita kirjautumismetodeja, erityisesti viimeisen version oletuksena olevan caching_sha2_password-sisäänkirjautumismetodin. Toinen parannusidea koskee komentovaihetta, johon voisi lisätä ominaisuuden, joka ilmoittaisi käyttäjälle, kuinka moneen riviin SQL-kysely on vaikuttanut. Viimeisenä kehitysideana MySQL-funktiolle on lisätä toiminnallisuuden, joka muuttaisi merkkijonona olevat rivitiedot siihen muotoon missä ne ovat tietokannassakin. Esimerkiksi kokonaisluvut, jotka on tallennettu listaan merkkijonona, muunnettaisiin kokonaislukumuotoon.

Microsoft SQL Server -funktioon voisi lisätä muita AsDb-kirjaston tarjoamia ominaisuuksia.

7 Yhteenveto

Insinöörityössä luotiin PLC-kirjasto, jonka avulla käyttäjä voi muodostaa yhteyden, lähettää SQL-kyselyitä sekä vastaanottaa vastauksia kahdesta tietokantahallintajärjestelmästä. Kommunikointi MySQL-hallintajärjestelmän kanssa on rakennettu määrittämällä ohjelmaan, kuinka MySQL Server/Client -protokollan paketteja luetaan ja rakennetaan. Tämän lisäksi ohjelma hyödyntää TCP-kirjastoa pakettien lähettämiseen sekä vastaanottamiseen.

Työ oli haastava ja moni asia on vaatinut sen syvää ymmärtämistä, joka teki siitä hyvin opettavaisen. Kaikista eniten ymmärrys ja osaaminen on kehittynyt laitteiden välisessä kommunikaatiossa sekä PLC-ohjelmoinnissa. Tämän lisäksi työssä pääsin tutustumaan SQL-ohjelmointiin sekä tietokantahallintajärjestelmien käyttöön. Haastavin, mutta siitä huolimatta mielenkiintoinen vaihe työssä oli salasanan salauksen toteutus. Siinä pääsi tutustumaan erilaisiin salausalgoritmeihin ja yleisesti siihen, miten nykypäivänä turvallisuus on toteutettu sisäänkirjautumisessa.

Asetetut tavoitteet ovat toteutuneet täysin. MySQL-funktio on rakennettu siten, että kommunikointi hallintajärjestelmän kanssa onnistuu tapauksesta huolimatta toisin, kun B&R:n tarjoamassa PLC-kirjastossa. Yksinkertaistettu Microsoft SQL Server -funktio konfiguroi ja suorittaa monet B&R:n kirjaston funktiot itse, jonka ansiosta kommunikoinnin rakentaminen käyttäjälle on äärimmäisen helppoa. Luotu PLC-kirjasto on helppokäyttöinen, jonka ansiosta kommunikointi hallintajärjestelmiin on helppo ja nopea toteuttaa.

Lähteet

A Beginner's PLC Overview. 2017. Verkkoaineisto. <https://www.automation.com/en-us/articles/2017/a-beginners-plc-overview-part-1-of-4-introduction>. Luettu 4.10.2021.

Ari Hovi. 2004. SQL-opas. E-kirja. Docendo.

Column Definition. Verkkoaineisto. <https://dev.mysql.com/doc/dev/mysql-server/latest/page_protocol_com_query_response_text_resultset_column_definition.html>. Luettu 16.11.2021.

COM_QUERY. Verkkoaineisto. <https://dev.mysql.com/doc/internals/en/com-query.html>. Luettu 8.11.2021.

COM_QUERY Response. Verkkoaineisto. <https://dev.mysql.com/doc/dev/mysql-server/latest/page_protocol_com_query_response.html>. Luettu 12.11.2021.

Connection Lifecycle. Verkkoaineisto. <https://dev.mysql.com/doc/dev/mysql-server/latest/page_protocol_connection_lifecycle.html>. Luettu 5.11.2021

Connection Phase. Verkkoaineisto. <https://dev.mysql.com/doc/dev/mysql-server/latest/page_protocol_connection_phase.html>. Luettu 5.11.2021

Douglas E. Comer. 2000. TCP/IP. Helsinki: Edita 2002.

EOF_Packet. Verkkoaineisto. <https://dev.mysql.com/doc/dev/mysql-server/latest/page_protocol_basic_eof_packet.html>. Luettu 4.11.2021.

ERR_Packet. Verkkoaineisto. <https://dev.mysql.com/doc/dev/mysql-server/latest/page_protocol_basic_err_packet.html>. Luettu 4.11.2021.

Integer Types. Verkkoaineisto. <https://dev.mysql.com/doc/dev/mysql-server/latest/page_protocol_basic_dt_integers.html>. Luettu 4.11.2021.

Kelechava, Brad. 2018. The SQL Standard – ISO/IEC 9075:2016. Verkkoaineisto. <https://blog.ansi.org/2018/10/sql-standard-iso-iec-9075-2016-ansi-x3-135/>. Luettu 23.11.2021.

MySQL Glossary. Verkkoaineisto. <https://dev.mysql.com/doc/refman/8.0/en/glossary.html#glos_dcl>. Luettu 22.11.2021.

MySQL Packets. Verkkoaineisto. <https://dev.mysql.com/doc/dev/mysql-server/latest/page_protocol_basic_packets.html>. Luettu 4.11.2021.

Native Authentication. Verkkoaineisto. <https://dev.mysql.com/doc/dev/mysql-server/latest/page_protocol_connection_phase_authentication_methods_native_password_authentication.html>. Luettu 8.11.2021.

OK_Packet. Verkkoaineisto. <https://dev.mysql.com/doc/dev/mysql-server/latest/page_protocol_basic_ok_packet.html>. Luettu 4.11.2021

Programming. Verkkoaineisto. <https://www.br-automation.com/en-gb/products/software/additional-information/programming>. Luettu 15.10.2021.

Protocol::HandshakeResponse. Verkkoaineisto. <https://dev.mysql.com/doc/dev/mysql-server/latest/page_protocol_connection_phase_packets_protocol_handshake_response.html>. Luettu 15.10.2021.

Protocol::HandshakeV10. Verkkoaineisto. <https://dev.mysql.com/doc/dev/mysql-server/latest/page_protocol_connection_phase_packets_protocol_handshake_v10.html>. Luettu 15.11.2021.

Relaatiotietokanta. Verkkoaineisto. <https://fi.wikipedia.org/wiki/Relaatiotietokanta>. Luettu 22.11.2021.

Steve Suehring. 2002. MySQL Bible. New York: Wiley Publishing, Inc.

String Types. Verkkoaineisto. <https://dev.mysql.com/doc/dev/mysql-server/latest/page_protocol_basic_dt_strings.html>. Luettu 4.11.2021.

Text Resultset. Verkkoaineisto. <https://dev.mysql.com/doc/dev/mysql-server/latest/page_protocol_com_query_response_text_resultset.html>. Luettu 12.11.2021.

What is a TCP/IP Socket Connection?. Verkkoaineisto. <https://www.ibm.com/docs/en/zvse/6.2?topic=SSB27H_6.2.0/fa2ti_what_is_socket_connection.html>. Luettu 5.11.2021