LAB University of Applied Sciences

Sähköautojen latauslaitteiden dynaaminen tehonrajoitusohjelmisto

LAB-ammattikorkeakoulu
Insinööri (AMK), Tieto- ja viestintätekniikka
2022
Aku Huotari

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Huotari, Aku	Opinnäytetyö, AMK	2022
	Sivumäärä	
	38	

Työn nimi

Sähköautojen latauslaitteiden dynaaminen tehonrajoitusohjelmisto

Tutkinto

Insinööri (AMK)

Toimeksiantajan nimi, titteli ja organisaatio

Rami Sivonen, R&D Manager, Kempower Oyj

Tiivistelmä

Työssä toteutettiin suurteholatureiden dynaaminen ohjausjärjestelmä, jolla säädetään latauslaitteiden lataustehoa kiinteistön kokonaissähkönkulutuksen mukaan. Työn tavoitteena on pystyä hyödyntämään kiinteistön sähköjärjestelmän kapasiteetti mahdollisimman tehokkaasti latauslaitteiden käyttöön. Toteutettu ohjelmisto kykenee säätämään Kempowerin suurteholatureiden lataustehoa. Kempower on suomalainen latauslaitteiden valmistaja.

Sähköautojen latauslaitteiden käyttöön erilaisissa käyttö ympäristöissä liittyy sähkönkulutuksen rajoituksia. Kiinteistöjen sähköjärjestelmien kapasiteetti on usein rajallinen, mikä saattaa rajoittaa latauslaitteiden käytössä olevaa sähkövirtaa. Kiinteistöissä saatetaan myös haluta säästää sähkönkulutuksessa. Verkkoon kytkettyjen latauslaitteiden lataustehoa voidaan ohjata OCPP-standardin mukaisilla Smart Charging-latausprofiileilla.

Työn aikana kehitetty ohjelmisto toteutettiin Amazon Web Services-pilvilaskenta-alustalle. Ohjelmiston toteuttamisessa hyödynnettävät ohjelmistoteknologiat olivat Node.js, TypeScript ja Serverless Framework. Ohjelmisto on toteutettu osaksi Kempowerin ChargEye SaaS-palvelua. Ohjelmiston mahdollinen jatkokehitys tapahtuu Kempowerin tuotekehitysyksikössä.

Asiasanat

pilvilaskenta, tehorajoitus, kuormanhallinta, latauslaitteisto, suurteholaturi

Abstract

Author(s)	Type of Publication	Published
Huotari, Aku	Thesis, UAS	2022
	Number of Pages	
	38	

Title of Publication

Dynamic load balancing for EV chargers with cloud computing

Name of Degree

Engineer (UAS)

Name, title and organization of the client

Rami Sivonen, R&D Manager, Kempower Oyj

Abstract

The aim was to implement a dynamic control system for high-power chargers, which adjusts the charging power of the charging devices according to the site's total electricity consumption. The goal is to utilize the capacity of the site's electrical system as efficiently as possible for the use of charging devices. The implemented software can adjust the charging power of Kempower's high-power chargers. Kempower is a Finnish charging equipment manufacturer.

Many environments of charging devices have different restrictions for electricity consumption. The capacity of electrical systems is often limited, which may limit the electrical current available for charging devices. Sites may also want to save on electricity consumption. The charging power of charging devices connected to the network can be controlled with Smart Charging profiles according to the OCPP standard.

The developed software was implemented on the Amazon Web Services cloud computing platform. The software technologies used in the implementation of the software were Node.js, TypeScript and Serverless Framework. The software has been implemented as part of Kempower's SaaS service ChargEye. Possible further development of the software takes place in Kempower's product development unit.

Keywords

cloud computing, power limitation, load management, charging equipment, high-power charger

Sisällys

1 Johdanto

Maailman autokanta on tällä hetkellä sähköistymässä kovaa vauhtia. Pelkästään Suomessa rekisteröitiin vuonna 2021 yli 10 000 uutta sähköhenkilöautoa (Autoalan tiedotuskeskus 2022). Sähköisten linja-autojen määrä Suomessa kasvoi kolminkertaiseksi ja myös sähköisiä kuorma-autoja voi nähdä liikenteessä. Myös ladattavien hybridien määrä liikenteessä on kasvanut viime vuosina. Sähköautojen lisääntyessä myös latausmahdollisuuksien kysyntä kasvaa. Pikalatausasemia pystytetään kovaa vauhtia ympäri maailmaa, jotta sujuva matkustaminen sähköautoilla olisi mahdollista. Tätä tukee esimerkiksi EU:n liikenneneuvosto esittämä asetus, joka vaatisi sähköautojen latausaseman 60 kilometrin välein kaikille pääväylille Euroopan Unionin alueella (Yle 2022).

Sähköautojen pikalatausasemilla käytettävät suurteholaturit ovat yleistymässä. Suurteholaturit pystyvät lataamaan sähköauton akun täyteen noin 30-60 minuutissa riippuen akun koosta. Tällä hetkellä suurteholaturit tarjoavat parhaimmillaan 150-350 kilowattia latausteho, mikä tarkoittaa noin 3-7 -kertaista latausteho verrattuna aikaisempiin 50 kilowatin pikalatauslaitteisiin. Yhä tehokkaampia latureita kehitetään jatkuvasti, ja latauslaitteisiin erikoistuneella teollisuuden alalla toimivat yritykset ovatkin todellisessa kilpajuoksussa keskenään. Koska suurteholaturit ovat suhteellisen uusi ja vielä monelle tuntematon aihe, niihin liittyviä haasteita ja ratkaisuja syntyy jatkuvasti.

Yksi suurteholatureihin liittyvä haaste on käytettävissä oleva virta. Esimerkiksi 200 kilowatin teholla toimiva suurteholaturi vaatii käyttöönsä yli 300 ampeeria sähkövirtaa kolmella vaiheella. Suuri sähkövirta tarkoittaa suurta sähkönkulutusta ja vaatii suurelle virralle suunnitellun sähköjärjestelmän. Kun suurteholaturin lisäksi paljon sähköä kiinteistössä kuluttavat muutkin sähkölaitteet, kuten esimerkiksi LVI-laitteet, saattaa sähkönkulutus ajoittain kasvaa todella suureksi. Kiinteistön sähköliittymän on siis oltava mitoitettu kestämään suurta rasitusta. Sen lisäksi kiinteistön sähkösopimuksen haltija myös maksaa kaikesta kiinteistössä käytetystä sähköenergiasta.

Rajoittamalla kiinteistön sähköverkkoa käyttävien latureiden virtaa voidaan laskea kiinteistön kokonaissähkönkulutusta. Alhaisempi laturin käytössä oleva virta kuitenkin tarkoittaa pienempää lataustehoa, mikä puolestaan pidentää latausaikaa. Tämä on ristiriidassa suurteholaturilla tavoiteltavien hyötyjen kanssa, sillä suurteholaturi pyrkii nimenomaan pienentämään laturissa vietettävää aikaa. Tämän lisäksi kiinteistön sähkönkulutus on harvoin tasaista. Yleisesti päivisin kulutus on korkeampaa kuin yöllä. Lämmitykseen ja viilennykseen sähköä kuluu riippuen ulkolämpötilasta. Myös sähköauton latauslaitteet kuluttavat eri määrän sähköä riippuen sähköauton mallista, akun koosta ja esimerkiksi lämpötilasta. Lataustehon rajoittaminen riittävästi, muttei tarpeettoman paljon, saattaa osoittautua hankalaksi.

Tämän työn päämäärä on toteuttaa dynaaminen kuormanhallintajärjestelmä verkkoon kytketyille latauslaitteille. Kyseessä on automatisoitu sähköajoneuvojen latauslaitteiden hallintajärjestelmä, jolla pystytään hyödyntämään kiinteistön varaamaton sähkövirran kapasiteetti mahdollisimman tehokkaasti latauslaitteiden käyttöön. Työ painottuu suuria sähkövirtoja vaativiin suuritehoisiin tasavirtalatureihin eli suurteholatureihin ja niiden virrankäytön dynaamiseen ohjaukseen. Työssä tutkitaan erilaisia suurteholatureiden käyttökohteita ja kartoitetaan erilaisten käyttökohteiden vaatimuksia. Tämän lisäksi työssä käydään läpi verkkoon kytkettyjen latureiden tiedonsiirtoa. Työn menetelminä toimivat viitekehyksen tutkiminen sekä ohjelmiston toteuttaminen.

Tavoitteiden saavuttamiseksi suurteholatureille asetetaan dynaaminen tehorajoitus, jota hallitaan Amazon Web Services pilvilaskenta-alustan avulla. Tehorajoituksien päämäärä on pitää kiinteistön kokonaissähkönkulutus sallitun raja-arvon alapuolella. Kokonaiskulutuksesta kerätty data lähetetään AWS pilvilaskenta-alustalle, jossa tieto käsitellään algoritmien avulla. Tehorajoitus määritetään mittaamalla kiinteistön kokonaissähkönkulutusta ja vertaamalla sitä kiinteistön sähköliittymän raja-arvoihin.

Kempower on suomalainen latauslaitteiden valmistaja. Kempowerin omistaa Kemppi-Group. Kemppi-Group on parhaiten tunnettu maailmanlaajuisesta hitsauslaitebrändistään Kempistä. Kempower-projekti aloitettiin ensimmäisen kerran 90-luvulla, kun kiinnostus muihin kuin hitsauslaitteiden tasasähköteholähteisiin lisääntyi. Kempower-projekti kuitenkin lakkautettiin vuonna 2009. Nykyinen Kempower perustettiin itsenäiseksi yritykseksi vuonna 2017 keskittyen täysin sähköautojen latausratkaisuihin. Kempower suunnittelee, valmistaa ja kaupallistaa latausratkaisuja ja palveluita erilaisissa olosuhteissa toimiville sähköajoneuvoille ja -koneille. Kempowerin latausratkaisut suunnitellaan ja valmistetaan Lahdessa.

2 Sähköajoneuvojen latauslaitteiden käyttöympäristöt ja latauslaitteiden etähallinta

2.1 Sähköajoneuvojen latauslaitteiden käyttöympäristöt ja lataustehon rajoitukset

Sähköajoneuvojen määrän kasvun myötä suurteholatureiden kysyntä on kasvanut valtavasti. Monet julkisia latauspisteitä tarjoavat tahot ovat suunnitelleet valtakunnallisia ja laajamittaisia latausverkostoja. Pelkästään vuoden 2022 ensimmäisen neljänneksen aikana yli 100 kilowatin lataustehoa tarjoavien latausasemien osuus kasvoi yli kolmannekseen kaikista tasavirtalatausta tarjoavista latausasemista. Tämänhetkisen arvion mukaan latausverkostoihin suuntautuvat investoinnit tulevat tulevaisuudessa painottumaan yhä enemmän suurteholatureihin. (Teknologian teollisuus 2022.)

Fossiilisia polttoaineita on totuttu tankkaamaan aina huoltoasemalla. Sähköisten ajoneuvojen kohdalla tilanne on toinen. Sähköajoneuvoa pystyy lataamaan kaikkialla, missä sähköä on saatavilla riittävästi. Lataus onnistuu huoltoaseman lisäksi esimerkiksi kotona, toimistolla tai ravintolan tai ostoskeskuksen parkkipaikalla. Julkisessa käytössä olevia latauslaitteita löytyy siis hyvin erilaisista käyttöympäristöistä. (EVBox.)

Erilaisten sähköajoneuvojen määrä on kasvussa myös julkisen liikenteen ja raskaan liikenteen osalta. Sähköistymisen myötä sujuva liikenne sähköajoneuvoilla tulee vaatimaan paljon erilaisia latausmahdollisuuksia. Etenkin raskas liikenne tarvitsee kapasiteetiltaan monenlaisia latauspisteitä, jotta sähköisen raskaan liikenteen toiminta olisi sujuvaa. Suuren kapasiteetin akustolla varustetut linja- ja kuorma-autot pyritään lataamaan varikolla. Joitakin sähköisiä linja-autoja pystytään lataamaan auton yläpuolelle sijoitetulla laturilla esimerkiksi linja-autopysäkillä matkustajia odottaessa. (Elenia 2022.)

Erilaiset käyttöympäristöt asettavat omat haasteensa ja rajoituksensa suurteholatureiden sähkönkulutukselle. Latauslaitteen fyysinen sijainti saattaa asettaa sähkönkulutukselle joitakin rajoitteita. Esimerkiksi kiinteistön sähköliittymä ja kiinteistön muu sähkönkulutus saattavat rajoittaa käytettävissä olevan sähkövirran määrää. Latauslaitteen etäisyys sähkökeskukselle on niin suuri, että kaapeloinnin mitoitus on haastavaa ja lataustehoa joudutaan tästä syystä rajoittamaan. Kiinteistössä saatetaan myös esimerkiksi haluta säästää sähkönkulutuksessa.

Suurteholaturit vaativat käyttöönsä paljon sähkövirtaa, jotta ne pystyvät tarjoamaan suuria lataustehoja. Latauslaitteille asetetaan joskus tehorajoituksia erilaisista syistä. Latauslaitteiden rajoitukset luodaan vastaamaan käyttöympäristön asettamia ehtoja sähkönkulutukselle. Latauslaitteen tehon rajoittamisessa oleellisinta on, että laitteen käytölle asetetut sähkönkulutuksen rajoitukset eivät ylity. Latauslaitteen käytössä olevaa sähkövirtaa pyritään

kuitenkin rajoittamaan niin, että latausteho ei laske tarpeettoman paljon vaaditun tason alapuolelle.

2.1.1 Henkilöautojen latausympäristöt

Sähköisten henkilöautojen lataus tapahtuu usein kotona. Lataukset ajoittuvat pääasiassa yöhön, ja lataussessiot venyvät pitkiksi. Latauslaitteiden teho täytyy mitoittaa käyttäjien tarpeiden ja kiinteistön sähköliittymän mukaisesti. Omakotitaloissa tarve on usein vain yhdelle tai kahdelle latauslaitteelle ja lataustehot pysyvät suhteellisen pieninä. Pientalon sähköliittymän koko on yleensä pieni, joten latausjärjestelmän hankinta saattaa vaatia sähköpääkeskuksen muutos- tai laajennustöitä. Sulakkeiden suojaamiseksi voidaan myös rajoittaa lataustehoa. (Elenia 2022.)

Myös taloyhtiö voi rakentaa latauspaikkoja ja hankkia latauslaitteita. Laitteita hankitaan joko tarpeen mukaan tai kerralla kaikille autopaikoille. Yleensä helpointa on pitää latauspisteiden määrä tarpeeksi pienenä, jotta voidaan välttyä kiinteistönsähköjärjestelmän muutoksilta. Tehorajoitukset ja kuormanhallinta ovat hyvä apu myös taloyhtiöiden sulakkeiden suojaamiseksi. (Elenia 2022.)

Julkisien latauspisteiden määrä on kasvanut viime vuosina. Latauspisteet painottuvat tällä hetkellä eteläiseen Suomeen ja suurimpiin kaupunkeihin. Latauspisteitä on sijoitettu esimerkiksi huoltoasemille, kauppakeskuksien ja kauppojen yhteyteen sekä joidenkin yrityksien kiinteistöihin. Julkisilla latauspaikoilla osa latauslaitteista on suurteholatureita, jotka mahdollistavat nopean latauksen. Suurteholaturit on yleensä sijoitettu huoltoasemien ja kauppojen yhteyteen. (Traficom 2021.)

Julkisten tilojen ja yrityksien kiinteistöjen yhteydessä olevien latureiden virrankäyttöön liittyy erilaisia rajoittavia tekijöitä. Kiinteistön sulakkeet ovat tässäkin kategoriassa suurin yksittäinen rajoitus. Isojen kiinteistöjen sähköliittymät ovat suurempia kuin pien- ja kerrostalojen, sillä myös sähkönkulutusta on enemmän. Esimerkiksi päivittäistavarakauppojen kiinteistöissä kuormitusta lisäävät muun muassa valaistus, pakastimet ja jääkaapit, LVI-laitteet sekä kiinteistöautomaatio. Tällaisten kiinteistöjen sähkönkulutus on myös erittäin vaihtelevaa. Päivisin kuormitus on yleensä paljon suurempaa kuin öisin, ja esimerkiksi ulkolämpötilat ja vuodenajat vaikuttavat kiinteistöjen sähkönkulutukseen. Lisäksi kiinteistössä saatetaan haluta rajoittaa kokonaiskulutusta esimerkiksi taloudellisista syistä. Julkisien tilojen ja yrityksien kiinteistöjen yhteydessä olevien suurteholatureiden virrankäyttöä saatetaan edellä mainituista syistä joutua rajoittamaan.

2.1.2 Linja-autojen latausympäristöt

Sähkölinja-autot ovat yleistyneet reilusti viime vuosina Suomessa ja maailmalla. Esimerkiksi Turun joukkoliikenteessä on yli 60 sähkölinja-autoa ja Helsingin seudun liikenteessä operoi 314 sähkölinja-autoa vuoden 2022 loppuun mennessä. Liikennöintisopimuksien uusiutumisen myötä sähkölinja-autoja voidaan olettaa ilmestyvän katukuvaan vielä enemmän tulevina vuosina. (Yle 2022.)

Yksi sähkölinja-autojen lataustavoista on niin sanottu päätepysäkkilataus. Päätepysäkkilataus mahdollistaa latauksen päivän aikana ajettavan linjan varrella. Päätepysäkkilatauksessa latausasemat on sijoitettu yleensä linjan päätepysäkeille. Linjan varrella sijaitsevat latauslaitteet ovat yleensä yli 300 kilowatin suurteholatureita, jotka hyödyntävät virroitinlatausta. Virroitinlatauksella tarkoitetaan linja-auton yläpuolelta tapahtuvaa latausta pantografin avulla. Virroitinlataus pystytään suorittamaan latauslaitteella tai raitiovaunun johdinverkosta. (Liikennevirasto 2017.)

Linja-autovarikoilla lataukset suoritetaan pääsääntöisesti kaapelilatauspisteillä. Linja-autot viettävät varikolla pidempiä aikoja kuin linjalla, joten pienemmät lataustehot ovat mahdollisia. Varikoiden latauslaitteistot on kuitenkin mitoitettu hyvin kattavasti. Varikoilla suositaan usein matalaa lataustehoa kustannuksien säästämiseksi. (Liikennevirasto 2017.)

Sähkölinja-autojen pikalatauspisteet vaativat tarpeeksi suuren sähköliittymän. Sähkölinja-autojen pikalataus tapahtuu 400-800 voltin tasajännitteellä yli 300 kilowatin teholla. Esimerkiksi 630 ampeerin sulakkeilla varustettu kolmevaiheliittymä pystyy syöttämään tarpeeksi tehoa kahdelle tai kolmelle pikalatauslaitteelle alennetulla latausteholla. Laitteiden määrän kasvaessa on joko kasvatettava sähköliittymän kapasiteettia tai rajoitettava lataustehoa.

Lataustehoa kuluttaa akuston lataamisen lisäksi myös sähkölinja-autojen esilämmitys. Esilämmitys tapahtuu saman latauslaitteiston avulla, jolla linja-auton akku ladataan. Tämä korostuu etenkin Suomessa, jossa lämpötilat laskevat talvisin pakkasen puolelle. Sisätilojen lämmitys kylminä talvipäivinä vaatii paljon energiaa, joten akun varaus kuluu nopeammin, mikä puolestaan lyhentää toimintamatkaa. Myös koneiston kylmät laakerit vaativat enemmän energiaa liikkuakseen. Esilämmityksen ansiosta linja-auto kuluttaa vähemmän akun varausta lämmitykseen, joten linja-auton toimintamatka kasvaa. Lämpimissä maissa esilämmityksen sijasta voidaan linja-auton sisätiloja viilentää etukäteen.

Oleellista linja-autojen latauksessa on, että linja-autot pääsevät lähtemään linjalle ajoissa täydellä akun varauksella. Varikoilla saattaa öisin olla latauksessa monta linja-autoa saman aikaisesti. Kun ladattavaa on paljon ja linja-autot esilämmitetään (tai viilennetään) ennen linjalle lähtöä, virrankäyttö saattaa nousta ajoittain todella korkeaksi. Suurilla varikoilla

pyritäänkin usein porrastamaan latauksia ja niiden tehoa aikataulujen mukaisesti, jotta virrankäyttö saataisiin pidettyä maltillisella tasolla.

2.1.3 Muiden sähköajoneuvojen latausympäristöt

Sähköhenkilö- ja linja-autojen lisäksi myös muiden sähköisten ajoneuvojen määrä on lisääntymässä. Raskaan liikenteen ajoneuvojen valmistajat ovat kehittäneet ja tuoneet markkinoille linja-autojen lisäksi myös sähkökuorma-autoja. Sähköinen raskas liikenne vaatii erilaisia latausjärjestelmiä ja oleellista raskaan liikenteen sähköistymisen kannalta onkin latausinfrastruktuurin kehittyminen. Myös esimerkiksi sähköllä toimivia veneitä ja näiden latauslaitteita on jo joissakin satamissa. Veneille on jo järjestetty pikalatausmahdollisuuksia ja yksi maailman ensimmäisistä veneiden suurteholatureista sijaitsee Norjan Florossa. Myös kaivosalalla on jo käytössä sähkökäyttöisiä kaivoskoneita ja erilaisia latausjärjestelmiä.

Sähkökuorma-autojen lataus voidaan hoitaa linja-autojen tapaan varikolla. Oleellista varikkolatauksessa on tietää reittisuunnitelma ja kuinka paljon reitin ajamiseen tarvitaan energiaa, sekä missä ajoneuvoa voi tarvittaessa ladata. Sähkökuorma-autoja voidaan tarpeen mukaan ladata integroidulla vaihtovirtalaturilla tai suuritehoisella tasavirtalaturilla. Pääsääntöisesti varikoilla suositaan pienempiä tehoja ja yön yli tapahtuvia latauksia. Suuritehoiset tasavirtalaturit mahdollistavat nopeamman latauksen, jolloin autoa on mahdollista ladata esimerkiksi kuljettajan tauon aikana. (Engdahl, 2021)

Sähköveneiden määrän lisääntymisen myötä sähköveneille tarkoitettu latausinfrastruktuuri on alkanut kehittymään satamissa. Veneilijöillä on jo ollut käytössään laiturisähköpisteitä, joiden yhteyteen voitaisiin lisätä sähköveneiden latausmahdollisuuksia. Satamien suurteholatureiden lataustehoa saatetaan joutua rajoittamaan esimerkiksi kaapeloinnin pitkien etäisyyksien vuoksi.

2.2 Verkkoon kytkettyjen latauslaitteiden etähallinta

Sähköautojen latauslaitteet voidaan jakaa verkkoon kytkettyihin ja verkkoon kytkemättömiin latauspisteisiin. Molemmat pystyvät lataamaan sähköautoon tarvittavan määrän energiaa, mutta verkkoon kytkettyjen ja verkkoon kytkemättömien latauslaitteiden kustannukset, saavutettavuus ja ylläpito eroavat toisistaan. Verkkoon kytketyt latauslaitteet yhdistetään Internetiin. Verkkoon kytkettyjen latauspisteiden etuja ovat esimerkiksi etähallintatyökalut ja näkyvyys latausverkostokarttapalveluissa. (Blink Charging Co. 2022.)

Latauslaitteiden kytkeminen verkkoon lisää Internet-järjestelmien avulla erilaisia latauslaitteen ominaisuuksia ja mahdollistaa yleisen toiminnallisuuksien valvonnan etänä.

Latauslaitteen Internet-yhteys mahdollistaa myös laitteella tapahtuvien lataustapahtumien tallentamisen sähköisiin järjestelmiin ja niiden valvomisen etänä. Internet-yhteyden avulla pystytään valvomaan esimerkiksi lataustapahtuman alkaessa käyttäjän tunnistautumista ja latauslaitteen sähkönkulutusta tietyn lataustapahtuman aikana. Internet-yhteys mahdollistaa myös latauksesta veloittamisen sähköisesti. (Blink Charging Co. 2022.)

Verkkoyhteydellä varustetun latauslaitteen etähallinta mahdollistaa myös erilaisia ylläpitotoiminnallisuuksia. Tällaisia ovat muun muassa latauslaitteen käynnistäminen uudelleen, laitteen poistaminen käytöstä, lataustapahtuman aloittaminen ja keskeyttäminen, latausprofiilin asettaminen ja ohjelmistoversion päivittäminen. Latauslaitteiden verkkoyhteydet mahdollistavat myös ohjelmistokehityksen kannalta tärkeiden lokitietojen keräämisen sähköisiin järjestelmiin, mikä helpottaa ja nopeuttaa erilaisten ongelmatilanteiden, kuten ohjelmointivirheiden, selvittämistä ja ratkaisua.

2.2.1 Open Charge Point Protocol

Open Charge Point Protocol, eli avoin latauspisteprotokolla, on standardi, joka tarjoaa yhtenäisen ratkaisun latauspisteiden ja taustajärjestelmien väliseen tiedonsiirtoon. Open Charge Point Protocol lyhennetään usein muotoon OCPP-standardi. OCPP mahdollistaa minkä tahansa taustajärjestelmän liittämisen minkä tahansa latauslaitteen kanssa riippumatta laitteen toimittajasta. Yhtenäinen standardi vähentää koordinointiongelmia ja on siten hyödyllinen koko sähköautomarkkinoille. (Open Charge Alliance.)

Open Charge Alliance on maailmanlaajuinen organisaatioiden yhteenliittymä, jonka tavoitteena on kehittää ja valvoa sähköajoneuvojen latausinfrastruktuurin viestintästandardeja. Open Charge Alliance hyödyntää sähköautojen latausinfrastruktuurin asiantuntijoiden tietoa ja kokemusta. Open Charge Point Protocol on keskeinen Open Charge Alliancen kehittämä standardi. OCPP-standardi on vapaasti kaikkien käytettävissä. Standardin tämänhetkisiä versioita ovat OCPP 1.5, OCPP 1.6, OCPP 2.0 ja OCPP 2.0.1. Standardin kehittäminen on jatkuvaa ja markkinoiden ohjaamaa, ja siinä otetaan huomioon teknologiset sekä liiketoiminnalliset tarpeet. (Open Charge Alliance.)

OCCP versio 1.5 julkaistiin vuonna 2013. Se pohjautuu Simple Object Access Protocol -tietoliikenneprotokollaan, joka toimii HTTP-protokollan yli. OCPP 1.5 mahdollistaa tiedonsiirron latauspisteen ja taustajärjestelmän välillä. Version keskeisiä ominaisuuksia ovat taustajärjestelmän ja latauspisteen välisen yhteyden luominen, tunnistautuminen lataustapahtuman alkaessa ja päättyessä, latauslaitteen vika- ja häiriötilojen ilmoittaminen, sekä diagnostiikka tietojen siirtäminen. OCPP 1.5 mahdollistaa myös latauspisteen varaamisen

ja latauspisteen konfigurointiasetusten muuttamisen taustajärjestelmän avulla. (Greenlots, 2018.)

OCPP versio 1.6 julkaistiin vuonna 2015. OCPP 1.6 tukee kaikkia OCPP 1.5 ominaisuuksia, mutta se ei ole taaksepäin yhteensopiva version 1.5 kanssa. OCPP 1.6 keskeisimmät erot edeltäjäänsä ovat sen uudet ominaisuudet. OCPP 1.6 pohjautuu JSON (JavaScript Object Notation) -tiedostoformaattiin, joka toimii WebSocket -protokollan yli. OCPP 1.6 mahdollistaa laajempien diagnostiikkatietojen siirtämisen, sekä pitää sisällään enemmän latauspisteiden tila- ja käynnistysviestejä. Yksi uusista ominaisuuksista on Smart Charging, joka mahdollistaa latauslaitteen käytössä olevaan sähkövirtaan tai lataustehoon vaikuttamisen taustajärjestelmän avulla. Muita uusia ominaisuuksia ovat muun muassa latauslaitteen ohjelmistoversion hallinta ja paikallisten valtuutusluetteloiden hallinta. (Open Charge Alliance, 2017.)

OCPP versio 2.0 julkaistiin vuonna 2018. OCCP 2.0 sisältää ominaisuuksia, jotka on kehitetty kasvavan sähköautojen latausinfrastruktuurin ja markkinoiden tarpeisiin. Merkittäviä uusia ominaisuuksia ovat muun muassa kehittyneempi lataustapahtumatietojen käsittely, kehittyneempi tietoturva ja Plug & Charge standardien sisällytys. Versio 2.0 sisältää myös laajemmat Smart Charging ominaisuudet kuin versio 1.6. (Greenlots, 2018.)

OCPP Versio 2.0.1. julkaistiin vuonna 2020. Kun OCPP 2.0 oli julkaistu, siinä havaittiin joitakin ongelmia, jotka on korjattu versiossa 2.0.1. Ongelmien korjaaminen vaati protokollan koneellisesti luettaviin skeemojen määritelmiin muutoksia, joten versio 2.0.1 ei ole taaksepäin yhteensopiva version 2.0 kanssa. Yhteensopivuusongelmien ja hämmennyksen vähentämiseksi versio on nimetty muotoon OCPI 2.0.1. (Open Charge Alliance, 2020.)

2.2.2 Smart Charging ja latausprofiilit

Smart Charging -ominaisuus esiteltiin ensimmäisen kerran OCPP versiossa 1.6. Smart Charging mahdollistaa tietyn lataustapahtuman tai koko latauslaitteen lataustehon muuttamisen taustajärjestelmän avulla. Taustajärjestelmä pystyy lähettämään latauslaitteelle latausominaisuuksia ja aikatauluja, jotka kuvastavat lataustapahtumaa ja tehon tai virran määrää, joka voidaan toimittaa tietyllä aikavälillä. Latauslaitteen käyttäjällä ei ole aktiivista roolia latausprofiilin määrittelyssä. (Orcioni, S & Conti, M. 2020.)

OCPP versiossa 2.0 esitelty Extended Smart Charging -ominaisuus laajentaa aiemmin OCPP 1.6 versiossa esiteltyä ominaisuutta. Extended Smart Charging pyrkii optimoimaan energianhallintaa ottaen huomioon energiantoimittajan tai esimerkiksi aurinkopaneeleista saatavan energian rajoitukset. Älykäs hallinta on saavutettu joustavammalla latausprofiililla. Latausprofiili voidaan asettaa latauslaitteelle lähettämällä ”SetChargingProfileRequest” -

pyyntö, joka on määritelty OCPP-standardissa. Latausprofiili voidaan asettaa lataustapahtuman alkaessa, ”RequestStartTransaction” -pyynnön yhteydessä, lataustapahtuman aikana aktiivisen latausprofiilin korvaamiseksi tai erillisenä viestinä ilman lataustapahtumaa. (Orcioni, S & Conti, M. 2020.)

Latauslaitteelle voidaan valmistajasta riippuen asettaa yksi tai useampi samanaikainen latausprofiili. Latausprofiili voi koskea joko yhtä lataustapahtumaa tai latauslaitetta. Mikäli latauslaitteelle on asetettu useampi samanaikainen latausprofiili, niin profiilin sisältävän viestin ”stackLevel” -kenttä määrittää kyseisen profiilin prioriteetin. Kuvassa 1 kuvataan viestin JSON-sisältöä, jolla asetetaan 20 kilowatin rajoituksen sisältävä latausprofiili latauslaitteelle. Tehorajoitus ilmoitetaan kentässä ”limit” ja yksikkö kentässä ”chargingRateUnit”.

```
{
    "id": 1000,
    "stackLevel": 1,
    "chargingProfileKind": "Relative",
    "chargingProfilePurpose": "TxDefaultProfile",
    "chargingSchedule": [
        {
            "id": 1001,
            "chargingRateUnit": "W",
            "chargingSchedulePeriod": [
                {
                    "startPeriod": 0,
                    "limit": 20000
                }
            ]
        }
    ]
}
```


Kuva 1. Latausprofiilin sisältävä JSON-viesti

Latausprofiilit mahdollistavat latauslaitteiden energiankulutuksen säätelyn. Säätely voidaan toteuttaa esimerkiksi erilaisilla algoritmeilla, jotka perustuvat lähtötietoihin ja ennalta määritettyyn logiikkaan. Lataus voidaan esimerkiksi toteuttaa pienellä teholla silloin, kun kiinteistön sähköliittymä rajoittaa virrankäyttöä tai sähköä ei jostain muusta syystä ole saatavilla tarpeeksi. Latausprofiilit mahdollistavat myös lataustapahtuman keston määrittelemisen ennalta, jolloin tehoa rajoitetaan niin, että haluttu akun varaustaso saavutetaan juuri ennen lataustapahtuman loppua.

3 Ohjelmistoarkkitehtuuri ja ohjelmointiympäristöt

3.1 Vaatimusmäärittely

Tavoitteena on toteuttaa dynaaminen kuormanhallintajärjestelmä verkkoon kytketyille latauslaitteille. Kyseessä on sähköajoneuvojen latauslaitteiden hallintajärjestelmä, jolla pystytään hyödyntämään kiinteistön varaamaton sähkövirran kapasiteetti mahdollisimman tehokkaasti latauslaitteiden käyttöön. Tavoitteen saavuttamiseksi luodaan ohjelmisto, joka lähettää latauslaitteille latausprofiileja, jotka sisältävät sähkövirrankäytön rajoituksia. Kiinteistön sähkönkulutus mitataan sähköpääkeskuksessa, jonka jälkeen mittaustulokset lähetetään rajapintojen ja tiedonsiirtokanavien avulla ohjelmistolle. Päämääränä on maksimoida latauslaitteen latausteho niin, että kiinteistön sähkönkulutuksen rajoitukset, kuten sähköliittymän rajoitukset, eivät ylity.

Dynaamisen kuormanhallinnan toiminnassa oleellista on, että latauslaiteiden virrankäyttöä säädetään kiinteistön kokonaissähkövirrankulutuksen perusteella. Rajoituksena kokonaisvirrankäytölle voi toimia esimerkiksi kiinteistön sulakkeiden koko. Kiinteistön kokonaiskulutuksen kasvaessa latauslaitteiden lataustehoa rajoitetaan, jotta kiinteistön kokonaiskulutus saadaan laskemaan. Vastaavasti latauslaitteiden lataustehoa voidaan nostaa silloin, kun kiinteistön kokonaisvirrankäyttö on sallittujen raja-arvojen alapuolella.

Dynaaminen kuormanhallintajärjestelmä voidaan konfiguroida asettamalla siihen liittyviä raja-arvoja. Merkittävin konfiguroitava raja-arvo on kiinteistön suurin mahdollinen virrankäytön määrä ampeereina. Muita konfiguroitavia arvoja on esimerkiksi pienin yhtä latauslaitetta kohden oleva sähkövirran määrä ampeereina ja latauslaitteen käytössä olevan virran määrä silloin, kun latauslaitteella ei ole yhteyttä taustajärjestelmään. Konfiguroitavat raja-arvot ovat avainasemassa toiminnallisen kokonaisuuden kannalta, sillä yhdessä kulutusmittaustietojen kanssa ne toimivat lähtötietoina lataustehoja laskevalle algoritmille.

3.1.1 Ohjelmisto

Dynaaminen kuormanhallinta toteutetaan pilvilaskenta-alustalla toimivalla ohjelmistolla. Ohjelmisto sisältää erilaisia mikropalveluja, kuten tiedonsiirtokanavan IoT-laitteelta pilvilaskenta-alustalle, algoritmin sisältävän lambdafunktion, sekä ohjelmiston eri osien välisen tiedonsiirron mahdollistavia rajapintoja. Ohjelmisto toteutetaan osaksi Kempowerin ChargEye -palvelua, ja ohjelmiston toteutuksessa hyödynnetään mahdollisimman tehokkaasti ennestään olemassa olevia ChargEye:n ominaisuuksia.

Ohjelmiston suoritusympäristönä käytetään Node.js avoimen lähdekoodin ympäristöä, joka toimii JavaScript-moottorilla. Ohjelmointikielinä toimivat TypeScript ja YAML. Ohjelmistoa

suoritetaan Amazon Web Services -pilvilaskenta-alustalla. Ohjelmisto toteutetaan Serverless-arkkitehtuurin, eli palvelimettoman tiedonkäsittelyn, periaatteita noudattaen, jossa taustajärjestelmän ohjelmiston suorittamiseen tarvitsema palvelinkapasiteetti otetaan käyttöön tarpeen mukaan. Ohjelmiston arkkitehtuuri perustuu mikropalveluihin, jotka muodostavat toiminnallisen kokonaisuuden. Mikropalvelut tallennetaan Amazon S3 (Simple Storage Service) -palveluun.

Ohjelmiston tulee skaalautua käyttötarpeen mukaan, jotta ohjelmisto pysyy toiminnallisena käyttökohteiden määrän lisääntyessä. Ohjelmiston täytyy olla saatavilla vuorokauden ympäri. Mahdollisia häiriötilanteita tulee yrittää ennakoida ohjelmistoa kehittäessä. Ohjelmisto tullaan toteuttamaan Amazon Web Services -pilvilaskenta-alustan tarjoamien palveluiden avulla.

3.1.2 Haasteet

Dynaamisen kuormanhallinnan tulee ennen kaikkea noudattaa ympäristön asettamia sähkövirrankäytön rajoituksia. Tämän toteuttamiseen pilvilaskenta-alustalla toimivalla ohjelmistolla liittyy erilaisia haasteita. Dynaamisen kuormanhallinnan on pystyttävä reagoimaan kiinteistön virrankulutuksen nousuihin mahdollisimman nopeasti ja tehokkaasti. Dynaamisen kuormanhallinnan tulee toimia aukottomasti niin, että ympäristön virrankäytön rajoituksia noudatetaan myös rajatapauksissa. Tällaisina rajatapauksina voidaan pitää esimerkiksi yhteysongelmia latauslaitteen tai sähkövirran määrää mittaavaan laitteen kanssa.

Tiedon siirtäminen sähkökeskuksessa sijaitsevalta mittalaitteelta on tärkeä osa toiminnallisuutta. Latauslaitteiden käytössä olevan sähkövirran määrää säädetään mittalaitteen lähettämien tietojen perusteella. Oleellista on, että mittalaitteelta lähetetty tieto pystytään saamaan taustajärjestelmälle mahdollisimman nopeasti, jotta pystytään reagoimaan viimeisimpiin mittaustuloksiin. Ohjelmiston on myös pystyttävä käsittelemään sähkönkokonaiskulutusta kuvaava tieto nopeasti. Jos esimerkiksi nähdään sähkönkulutuksen olevan nousussa, niin lataustehoa on rajoitettava välittömästi. Tiedon tulee kulkea jatkuvasti ja tiedon oikeellisuus tulee todentaa ennen kuin tietoa käsitellään algoritmin avulla.

Ohjelmiston toteutuksessa tulee huomioida mahdolliset tiedonsiirto-ongelmat eri laitteiden ja taustajärjestelmän välillä. Mikäli mittalaitteelta ei saada ajantasaista tietoa sen hetkisestä kokonaisvirrankulutuksesta, niin latauslaitteiden käytössä olevaa sähkövirran määrää ei pystytä määrittämään dynaamisesti. Tällaisia tilanteita varten on oltava olemassa ennalta määritelty vikasietotila. Toinen mahdollinen rajatapaus on latauslaitteen ja taustajärjestelmän väliset tiedonsiirto-ongelmat. Mikäli latauslaitteeseen ei saada yhteyttä, on silloin

latauslaitteella oltava valmiina konfiguraatio, joka määrittää tällaisessa tilanteessa suurimman sallitun sähkövirran määrän.

3.2 Ohjelmointiteknologiat

Työhön on valittu ChargEye:ssa aiemmin käytettyjä ohjelmointiteknologioita. Valitut teknologiat ovat tunnettuja ja laajasti käytettyjä verkkoapplikaatioiden kehityksessä. Tunnettujen ja paljon käytettyjen ohjelmointiteknologioiden etuna voidaan pitää niiden tuen ja saatavuuden jatkuvuutta tulevaisuudessa.

JavaScript on erittäin suosittu ohjelmointikieli, johon viitataan usein ”suorita missä tapahansa” -tyyppisellä ilmauksella. JavaScript luotiin alun perin käytettäväksi verkkosivujen ohjelmoinnissa, mutta nykyään JavaScriptiä käytetään kokonaisten ohjelmistopinojen kehittämiseen. TypeScript on JavaScript pohjainen ohjelmointikieli, joka sisältää syntaksin eri tyypityksille. Ylimääräinen syntaksi mahdollistaa paremman integraation ohjelmointieditorin kanssa, mikä helpottaa ja nopeuttaa ohjelmistokehitystä. TypeScript kootaan aina ennen suorittamista JavaScript muotoon. (TypeScript, 2022.)

Ohjelmiston suoritusympäristönä toimii avoimen lähdekoodin Node.js. Se tunnetaan suosittuna JavaScriptin suoritusympäristönä palvelimilla. Node.js toimii asynkronisesti yhdessä prosessissa. Node.js on suunniteltu optimoimaan suorituskykyä ja skaalautuvuutta verkkosovelluksissa, joissa on monia syöttö- ja ulostulotoimintoja, sekä verkkosovelluksissa, jotka toimivat reaaliajassa. Node.js on kevyt ohjelmointikehys, joka sisältää pienen määrän oletusmoduuleja. Moduuleja voi lisätä tarpeen tullen, mikä pitää ohjelmiston mahdollisimman suorituskykyisenä. Node.js yhteisö on kehittänyt tuhansia erilaisia kirjastoja, jotka ovat vapaasti kaikkien kehittäjien käytettävissä. (Node.js.)

Serverless tarkoittaa palvelimetonta tietojenkäsittelyä, jota voidaan hyödyntää pilvilaskenta-alustoilla. Palvelimeton tietojen käsittely tarkoittaa, että pilvilaskenta-alusta tarjoaa ohjelmistonsuorittamiseen vaaditun määrän palvelinkapasiteettia sen hetkisen kysynnän mukaan. Kehittäjien ei tarvitse olla tietoisia taustalla olevasta infrastruktuurista. Serverless-ohjelmisto voidaan toteuttaa käyttämällä Serverless-ohjelmointikehystä. Monet pilvilaskentapalveluja tarjoavat alustat tukevat ja tarjoavat palvelimettomia tietojenkäsittely palveluja. (Cloudflare.)

3.2.1 TypeScript

TypeScript on Microsoftin kehittämä ja ylläpitämä ohjelmointikieli ilmainen ja avoimen lähdekoodin ohjelmointikieli. TypeScriptin on suunnitellut C# -ohjelmointikielen suunnittelija

Anders Hejlsberg. TypeScript on sekä ohjelmointikieli, että joukko erilaisia työkaluja. TypeScript on käytännössä JavaScriptiä, joka sisältää enemmän ominaisuuksia.

JavaScript kehitettiin alun perin yksinkertaiseksi komentosarjakieleksi selaimille. JavaScriptiä odotettiin käytettävän verkkosivujen upotetuissa ohjelmakoodeissa. Kun aikaa kului, JavaScriptistä tuli entistä suositumpi ja kehittäjät alkoivat käyttää sitä verkkosivujen interaktiivisiin toimintoihin. Lisääntynyt JavaScriptin käyttö johti suoritusmoottorien optimoimiseen ja JavaScriptin ominaisuuksien lisäämiseen, mikä puolestaan teki JavaScriptistä entistä suositumman ohjelmointikielen. JavaScript keräsi niin paljon suosiota, että sitä alettiin käyttämään myös selainten ulkopuolella, kuten palvelimien toteuttamisessa Node.js:n avulla. Nykyään JavaScript on yksi suosituimmista ohjelmointikielistä ja sitä käytetään jopa kokonaisten ohjelmistopinojen luomiseen. (Microsoft)

TypeScript sisältää kaikki JavaScriptin ominaisuudet ja lisää kerroksen JavaScriptin ominaisuuksien päälle. TypeScript sisältää vahvan tyyppijärjestelmän ja staattisen tyypin tarkastuksen. TypeScript tarkastaa ohjelman virheiden varalta ennen ajoa arvojen tyyppien perusteella. Vertailun vuoksi JavaScript tarjoaa samat kieliprimitiivit, kuten merkkijonon ja numeron, mutta se ei tarkasta niiden määritystä. TypeScriptin suurin etu on, että se korostaa virheellisiä ohjelmakoodin määrityksiä, mikä vähentää bugien mahdollisuutta ohjelmaa suoritettaessa. (Microsoft.)

JavaScript sisältää valmiiksi joitakin primitiivityyppejä: boolean, bigint, null, number, string, symbol ja undefined. TypeScript laajentaa tätä luetteloa muutamalla tyypillä: any, unknown, never ja void. Tyypityksien määrittämiseksi on kaksi mahdollista syntaksia: interface ja type. Kehittäjän tulisi ensisijaisesti käyttää Interface-syntaksia tyypityksien määrittämiseen ja Type-syntaksia silloin, kun tarvitaan erityisiä ominaisuuksia. TypeScript mahdollistaa myös monimutkaisempien tyyppien luomisen yhdistämällä yksinkertaisia tyyppejä. (Microsoft.)

Kuvassa 2 havainnollistetaan TypeScriptin toimintaa. Kuvan funktiolle voidaan asettaa vain string-tyyppinen parametri ja funktio lupaa palauttaa aina number-tyyppisen arvon. Mikäli funktiota yritetään kutsua asettamaan arvo string-tyyppiselle muuttujalle, TypeScript huomaa virheellisen määrityksen ja ilmoittaa siitä. Virheellinen määritys näkyy kuvassa punaisella alleviivattuna.

```
function exampleFunction(exampleParam: string): number {
    const result: number = +exampleParam
    return result
}
const numberVariable: number = exampleFunction('123')
const stringVariable: string = exampleFunction('123')
```

Kuva 2. Esimerkki TypeScriptin toiminnasta

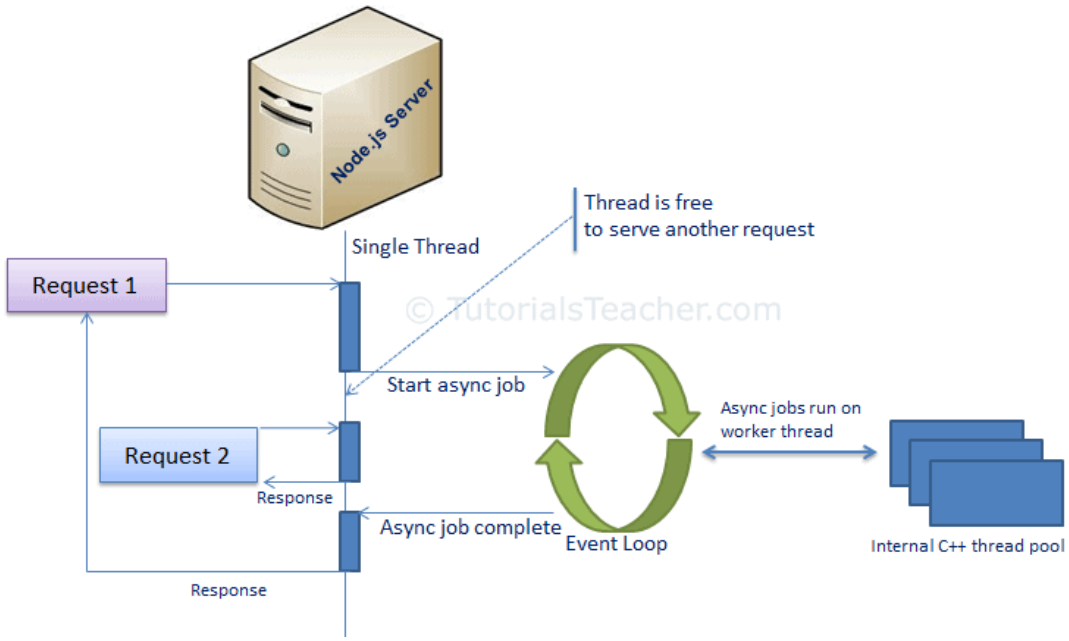
Kehittäjät käyttävät TypeScriptiä sen tyypintarkastuksen vuoksi erilaisista syistä. Jotkin kehittäjät haluavat niin sanotusti väljempää tyypitystä, joka auttaa validoimaan vain osan ohjelmasta. TypeScriptin oletuskäyttökokemus onkin, että tyypit ovat valinnaisia, eikä kaikkia mahdollisia tyhjiä tai määrittelemättömiä arvoja tarkasteta. Osa kehittäjistä haluaa kuitenkin, että TypeScript validoi koodia niin paljon kuin mahdollista. Tätä varten TypeScript tarjoaa erittäin tarkat validointiasetukset. Kehittäjälle tarkempi validointi saattaa tarkoittaa ylimääräistä työtä, mutta yleensä tarkempi validointi on pitkällä aikavälillä parempi vaihtoehto, koska se mahdollistaa tarkemman tarkastuksen ja työkalun. Uuden koodikannan tulisi aina käyttää samantasoisia tai tarkempia tarkkuusasetuksia. (Microsoft, 2022.)

3.2.2 Node.js

Node.js on avoimen lähdekoodin suoritusympäristö JavaScriptille selaimen ulkopuolella. Node.js käyttää Chromen V8 JavaScript-moottoria. Node.js on tapahtumapohjainen, asynkroninen ja järjestelmästä riippumaton suoritusympäristö, joka on suunniteltu skaalautuville palvelimella suoritettaville ohjelmistoille. Alun perin Node.js:n on luonut Ryan Dahl vuonna 2019. (TutorialsTeacher.com, 2022.)

Node.js toimii asynkronisesti. Se käyttää tapahtumasilmukaksi (Event Loop) kutsuttua toimintoa, joka varmistaa, että tietoa ei tarvitse kysyä jatkuvasti, vaan tietoa siirretään silloin, kun se on olemassa. Kun tapahtumasilmukka käsittelee asynkronisia tehtäviä, Node.js voi samaan aikaan jatkaa ohjelman suorittamista. Tämän vuoksi Node.js on optimaalinen silloin, kun suoritettavaa ei ole, sen sijaan, että se kyselisi tietoja jatkuvasti. Node.js on hyvä työkalu skaalautuvien ohjelmistojen rakentamiseen. Tapahtumasilmukka hallitsee viestintää varaamatta paljon muistia, mikä mahdollistaa ohjelmiston kehittämisen ilman, että kehittäjät joutuvat huolehtimaan liian monien kyselyiden aiheuttamasta ohjelmiston hidastumisesta. (ReadWrite, 2013.)

Kuvassa 3 havainnollistetaan Node.js:n ja sen tapahtumasilmukan toimintaa. Yksi säie käsittelee pyyntöjä ja asettaa ne tapahtumasilmukaan. Tapahtumasilmukka käsittelee pyynnöt sisäisten säikeiden avulla asynkronisesti. Pyyntöjä käsittelevä säie palauttaa vastauksen, kun pyyntö on käsitelty tapahtumasilmukassa.



Kuva 3. Node.js tapahtumasilmukan toiminta (TutorialsTeacher.com)

Node.js käyttää moduuleja, jotka ovat yksinkertaisia tai monimutkaisia toimintoja, joita voidaan kutsua ja käyttää koko Node.js-ohjelmassa. Moduulien toiminnallisuudet on järjestelty yhteen tai useampaan JavaScript -tiedostoon, joita voidaan kutsua viittaamalla haluttuun moduuliin. Jokainen Node.js moduuli toimii omassa kontekstissaan niin, että sen toiminta ei häiritse muita moduuleja. Node.js noudattaa CommonJS modules standardia, määrittelee JavaScriptin standardit web-palvelimelle, työpöydälle ja konsoliohjelmille. Node.js:n niin sanotut ydinmoduulit sisältävät sen vähimmäistoiminnot. Ydinmoduulit sisällytetään aina Node.js -ohjelmaan, ja ne ladataan automaattisesti aina ohjelman käynnistyessä. Kehittäjät pystyvät lisäämään moduuleja sen mukaan, kun niille tulee tarvetta. Tämä mahdollistaa ohjelmiston pitämisen mahdollisimman suorituskykyisenä. (TutorialsTeacher.com)

Node.js:n moduuleja hallitaan Node Package Manager (npm) -ohjelmistorekisterin avulla. Se on maailman suurin ohjelmistorekisteri, ja kehittäjät käyttävät sitä moduulien jakamiseen ja lainaamiseen. Myös moni organisaatio käyttää Node Package Manageria yksityisessä ohjelmistokehityksen hallinnassa. Node Package Manager koostuu kolmesta osasta: verkkosivusta, komentorivikäyttöliittymästä (CLI) ja rekisteristä. Kehittäjien näkökulmasta tärkein vuorovaikutus npm:n kanssa tapahtuu komentorivikäyttöliittymästä, jolla hallitaan

moduulien julkaisua ja latausta. Npm sisältää esimerkiksi moduulien versionhallinnan, joka helpottaa huomattavasti ohjelmiston päivittämistä. (Npm.)

3.2.3 Serverless

Serverless, eli palvelimeton tietojenkäsittely, on yksi pilvilaskennan palvelumuoto. Serverless-arkkitehtuurissa pilvilaskenta-alusta tarjoaa palvelinkapasiteettia ohjelmiston käyttöön tarpeen mukaan. Serverless-arkkitehtuurin myötä kehittäjien ei tarvitse miettiä taustalla olevaa palvelininfrastruktuuria. Serverless palveluiden käytössä on hyötyjä. Serverless-palveluiden käytöstä laskutetaan käytön mukaan, joten käyttäjien ei tarvitse maksaa kiinteitä kuluja käyttämättömästä palvelinkapasiteetista. Lisäksi ohjelmiston kehittämisen aloittaminen on nopeaa, ja kehittämisessä tarvittava ennakoinnin tarve on ohjelmiston skaalautuvuuden ansiosta pienempi. (Lintilä, 2017.)

Serverless eroaa muista pilvilaskentamalleista siinä, että pilvilaskenta-alustan tarjoaja vastaa sekä palvelininfrastruktuurin hallinnasta, että sovelluksien skaalautuvuudesta. Serverless-ohjelmistot otetaan käyttöön pilvilaskenta-alustalla säiliöissä, jotka käynnistyvät automaattisesti niitä kutsuttaessa. Pilvilaskenta-alustan asiakas maksaa palvelimettomasta tietojenkäsittelystä vain sen ajan, kun kutsuttua ohjelmaa suoritetaan. (Red Hat, 2022.)

Serverless voidaan jakaa kahteen ryhmään: Backend-as-a-Service (BaaS) ja Function-as-a-Service (FaaS). BaaS mahdollistaa kehittäjien pääsyn useisiin kolmannen osapuolen palveluihin ja sovelluksiin. Pilvilaskenta-alustalla voi olla tarjolla esimerkiksi todennuspalveluja, salauspalveluja tai tietokantoja. BaaS-tyyppisiä serverless-toimintoja kutsutaan yleensä ohjelmistorajapintojen kautta. Yleisimmin serverless viittaa kuitenkin FaaS-malliin. FaaS-tyyppiset serverless-ohjelmistot ajetaan säiliöissä, jotka ovat kokonaan pilvilaskenta-alustan hallussa. Suurimmilla pilvilaskenta-alustojen tarjoajilla on yksi tai useampi FaaS-alusta, kuten esimerkiksi Amazonilla AWS Lambda ja Microsoftilla Azure Functions. (Red Hat, 2022.)

Yksi tapa toteuttaa serverless-ohjelmisto on Serverless Framework. Serverless Framework mahdollistaa palvelimettoman tiedonsiirtoarkkitehtuurin mukaisten ohjelmistojen kehittämisen, käyttöönoton, vianmäärityksen ja suojauksen. Serverless Frameworkia voidaan käyttää avoimen lähdekoodin komentorivikäyttöliittymän avulla. Serverless Frameworkin auttaa kehittämään esimerkiksi AWS Lambda-funktioita ja toimintoja, sekä niiden tarvitsemia AWS infrastruktuuriresursseja. (Serverless, Inc.)

Ohjelmiston palveluinfrastruktuuri voidaan määrittää serverless.yml-tiedostossa, joka hallitsee lähes kaikkea Serverless Frameworkin avulla käyttöön otettavaa ohjelmistoa. Serverless.yml-tiedostossa määritellään kaikki itsenäiset mikropalvelut, kuten infrastruktuurikomponentit, joita Lambda-funktiot käyttävät, muista resursseista tulevat tapahtumalaukaisimet,

sekä rajapinnat ja niiden käyttöoikeudet. Serverless.yml sisältää myös pilvilaskenta-alustan sisäisten resurssien oikeudet käyttää muita resursseja ja ominaisuuksia. Mikäli käyttäjä haluaa palveluinfrastruktuurinsa määrittämiseen enemmän joustavuutta, sen määrittäminen onnistuu myös JSON, JavaScript tai TypeScriptin avulla. Kuvassa 4 on serverless.yml tiedosto, jossa on määritelty Lambda-funktio, jota voidaan kutsua HTTP API:n avulla. Kuvassa Lambdalle on myös määritetty ”someJwtAuthorizer” niminen tunnistautumista valvova resurssi, joka tarkastaa kutsujan lähettämän JSON Web Tokenin validiteetin. (Serverless, Inc.)

```
someFunction:
  handler: index.handler
  events:
    - httpApi:
        method: POST
        path: /some-post
        authorizer:
          name: someJwtAuthorizer
```

Kuva 4. Lambda-funktion määrittely serverless.yml-tiedostossa (Serverless, Inc.)

3.3 Amazon Web Services -pilvilaskenta-alusta

Pilvipalvelut ja niiden tarjoamat pilvilaskenta-alustat toimivat tilauspohjaisella toimintamallilla ja tarjoavat teknologioita erilaisiin käyttötarkoituksiin. Pilvilaskentaresursseista maksetaan kulutuksen mukaisesti, ja resurssien määrää voidaan nostaa tai laskea tarpeen mukaan. Pilvilaskennan avulla toteutetut ohjelmistot ovat skaalautuvia. Pilvipalveluiden teknologiat ovat helposti saatavilla, mikä nopeuttaa ohjelmistokehittämistä ja lisää ohjelmiston toiminnallisuuksien mahdollisuuksia. (Amazon Web Services, Inc.)

Amazon Web Services eli AWS on maailman laajimmin käytetty pilvipalvelu. AWS sisältää yli 200 erilaista palvelua, jotka ovat tarjolla palvelinkeskuksissa ympäri maailman. AWS tarjoamat palvelut sisältävät muun muassa pilvilaskennan infrastruktuuriteknologioita, kuten laskenta-, tallennus- ja tietokantaratkaisuja, teknologioita koneoppimiseen ja tekoälyyn, tietoalustoja ja analytiikkatyökaluja, sekä IoT-laitteiden palveluja. (Amazon Web Services, Inc.)

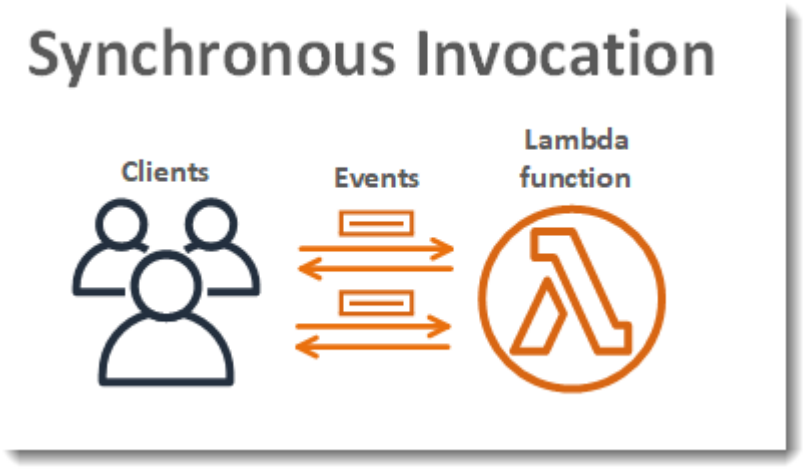
3.3.1 AWS Lambda

AWS Lambda on pilvilaskentapalvelu, jonka avulla voi suorittaa koodia ilman palvelinkapasiteetin varausta ja hallintaa. Lambda hallitsee kaikkia sen käyttämiä laskentaresursseja, kuten palvelimen ja käyttöjärjestelmän ylläpitoa, kapasiteetin varauksen ja automaattisen skaalauksen, sekä lokikirjoituksen. Lambdalla voi suorittaa erityyppisiä sovelluksia ja taustajärjestelmiä. Lambda pystyy suorittamaan esimerkiksi JavaScriptiä Node.js:n avulla. (Amazon Web Services, Inc.)

Lambda-funktiot ovat AWS Lambdan perusresurssi. Lambdaa käytettäessä ohjelmiston koodi järjestellään Lambda-funktioiksi. Lambda-funktioita suoritetaan vain pyydettäessä ja samaa funktiota pystytään suorittamaan samanaikaisesti monissa eri instansseissa. Funktion suorittaminen perustuu tapahtumaan, jolla funktiota on kutsuttu, ja tietoihin, joka funktiolle on kutsuttaessa annettu. (Amazon Web Services, Inc.)

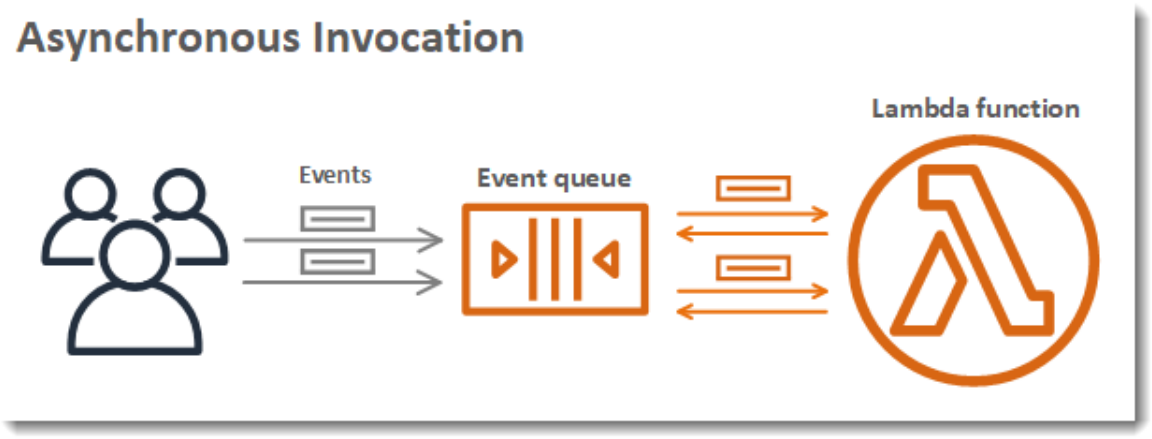
Lambda-funktioita voi kutsua esimerkiksi ohjelmointirajapinnoilla tai funktiokutsut voidaan sitoa tiettyihin tapahtumiin muissa AWS-palveluissa. Tällaisia Lambda-funktioiden kutsuja sanotaan triggereiksi. Kutsun yhteydessä Lambdalle lähetetään JSON-muotoinen dokumentti, joka sisältää kutsutapahtuman tiedot, jotka funktio käsittelee. Kutsutapahtuman tiedot muunnetaan tapahtumaobjektiksi, ja se toimitetaan suoritettavalle ohjelmalle. Lambda-funktioita voi kutsua joko synkronisesti tai asynkronisesti. (Amazon Web Services, Inc.)

Synkronisesti kutsuttaessa Lambda suorittaa funktion, ja kutsuja jää odottamaan vastausta. Kun funktio on valmis, Lambda palauttaa funktion vastauksen tiedot JSON-formaatissa. Lambda palauttaa myös tapahtuman suorittaneen Lambdan version ja statuskoodin. Kuvassa 5 havainnollistetaan Lambda-funktion synkronista kutsumista. (Amazon Web Services, Inc.)



Kuva 5. Lambda-funktion synkroninen kutsuminen (Amazon Web Services, Inc.)

Asynkronisesti kutsuttaessa Lambda-funktion vastausta ei jäädä odottamaan, vaan kutsutapahtuma lähetetään Lambdalle ja funktio suoritetaan. Monet AWS-palvelut, kuten Amazon Simple Storage Service (Amazon S3) ja Amazon Simple Notification Service (Amazon SNS) kutsuvat Lambdaa tällä tavalla. Asynkronisesti kutsuttaessa Lambda asettaa kutsutapahtumat jonoon ja palauttaa vastauksen onnistuneesta vastaanotosta. Erillinen prosessi lukee kutsutapahtumat jonosta ja välittää tiedot funktiolle. Kuvassa 6 havainnollistetaan Lambda-funktion asynkronista kutsumista. (Amazon Web Services, Inc.)



Kuva 6. Lambda-funktion asynkroninen kutsuminen (Amazon Web Services, Inc.)

Lambda-funktioille voidaan määrittää erilaisia suoritusrooleja, jotka myöntävät funktiolle luvan käyttää AWS-palveluita ja -resursseja. Roolit määritetään, kun funktio luodaan, ja Lambda käyttää roolia silloin, kun sitä kutsutaan. Määritetyt roolit voivat myöntää pääsyn esimerkiksi tiettyyn tietokantaan tai viestijonoon. Roolien määrittäminen on tietoturvan kannalta olennaista. Palveluihin ja resursseihin tulisi aina myöntää vain tarvittavat käyttöoikeudet. (Amazon Web Services, Inc.)

3.3.2 Amazon DynamoDB

AWS DynamoDB on NoSQL-tietokantapalvelu. DynamoDB:n avulla pystyy luomaan tietokantataulukoita, jotka pystyvät tallentamaan ja hakemaan minkä tahansa määrän tietoa. Tietokantataulukot pystyvät palvelemaan mitä tahansa määrää pyyntöliikennettä. Tietokantataulukot ovat skaalautuvia, mikä mahdollistaa kapasiteetin pienentämisen tai suurentamisen ilman palvelun estymistä tai suorituskyvyn heikkenemistä. (Amazon Web Services, Inc.)

AWS DynamoDB koostuu kolmesta ydinkomponentista: tietokantataulukoista, taulukoihin tallennettavista objekteista ja attribuuteista. DynamoDB tallentaa tietoa

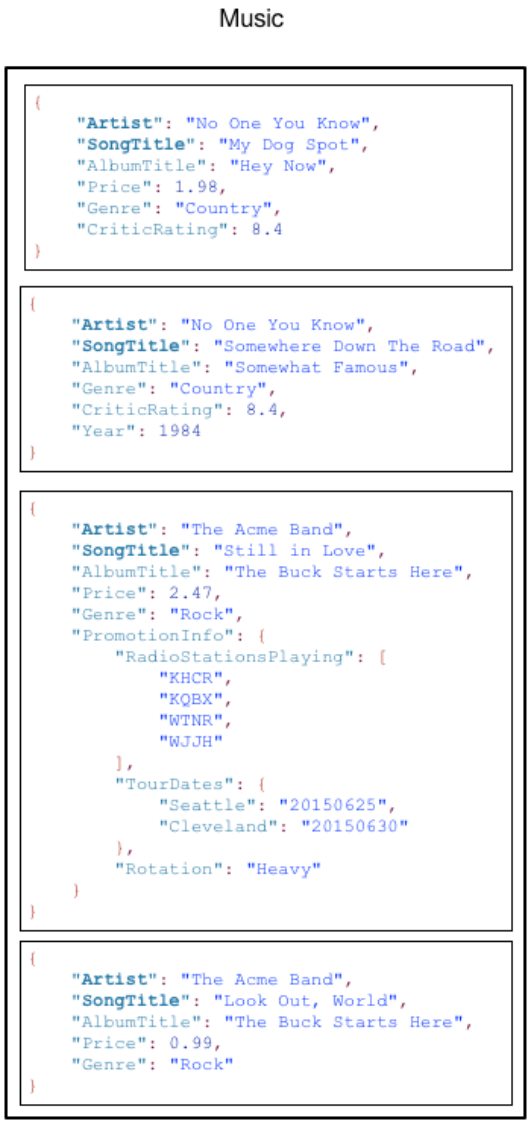
tietokantataulukoihin, jotka toimivat tietokokoelmina. Kuvassa 7 kuvataan tietokantataulukkoa nimeltä ”People”. Tietokantataulukkoon tallennettavat tiedot tallennetaan objekteina. Kuvassa 7 objektit kuvaavat henkilöitä, ja jokainen objekti sisältää tietyn henkilön tiedot. Tallennettavat objektit sisältävät attribuutteja, jotka sisältävät objektin yksilöiviä tietoja. Kuvassa 7 henkilöille on määritetty useita attribuutteja, kuten sukunimi, etunimi ja puhelinnumero. (Amazon Web Services, Inc.)

Kuva 7. Esimerkki DynamoDB:n tietokantataulusta (Amazon Web Services, Inc.)

Tietokantataulukolle määritetään taulukon ensisijainen avain, jonka avulla voidaan tunnistaa yksilöllisesti jokainen taulukon objekti. Kahdella objektilla ei voi olla samaa avaimen arvoa. DynamoDB tukee kahta erilaista ensisijaista avainta: partitioavainta sekä yhdistetystä partitio- ja lajitteluavaimesta muodostettua avainta. (Amazon Web Services, Inc.)

Jos tietokantataulukko käyttää vain partitioavainta, kahdella kohteella ei voi olla samaa partitioavaimen arvoa. Kuvassa 8 on henkilöobjekteilla partitioavain on ”PersonID”-niminen attribuutti. Objekti pystytään etsimään tietokantataulukosta suoraan kyseisen objektin partitioavaimen avulla. (Amazon Web Services, Inc.)

Partitio- ja lajitteluavaimen muodostama ensisijainen avain koostuu kahdesta attribuutista, jota kutsutaan yhdistetyksi ensisijaiseksi avaimeksi. Ensimmäinen attribuutti on partitioavain ja toinen attribuutti on lajitteluavain. Partitioavainta käytetään sisäisen hajautusfunktion syötteenä. Hajautusfunktion tulos määrittää partition, johon objekti tallennetaan. Kaikki objektit, jotka jakavat saman partitioavaimen arvon, tallennetaan yhdessä lajitteluavaimen arvon määrittämässä järjestyksessä. Kaikki objektit, joilla on sama partitioavaimen arvo, on oltava yksilöllinen lajitteluavaimen arvo. Kuvassa 8 kuvataan ”Music”-nimistä tietokantataulukkoa, jossa on käytetty yhdistettyä ensisijaista avainta. Taulukosta pystytään etsimään esimerkiksi yhden esittäjän kaikki kappaleet pelkällä ”Artist” -attribuutilla. Jos taulukosta halutaan etsiä tietty objekti, voidaan kyselyssä käyttää ”Artist” - ja ”SongTitle” -attribuuttien yhdistelmää.



Kuva 8. Yhdistettyä ensisijaista avainta käyttävä tietokantataulukko (Amazon Web Services, Inc.)

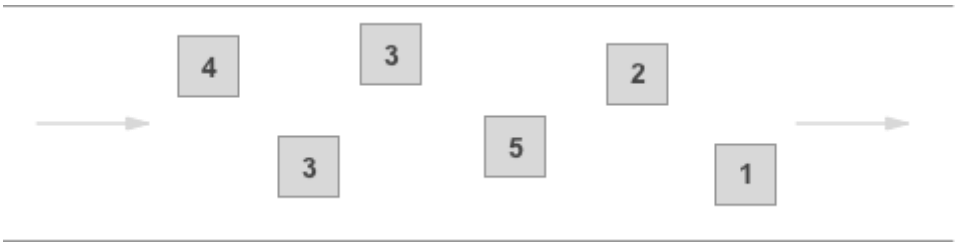
3.3.3 Amazon Simple Queue Service ja Amazon SNS

Amazon Simple Queue Service (SQS) ja Amazon Simple Notification Service (SNS) ovat AWS-pilvilaskenta-alustalla toimivia viestipalveluja, joita käytetään hajautettujen viestintäjärjestelmien toteuttamiseen. Amazon SQS ja Amazon SNS toimivat pilvilaskenta-alustalla käytettävien resurssien välissä viestinvälittäjinä. Niiden avulla voi lähettää, tallentaa ja vastaanottaa viestejä eri ohjelmistokomponenttien välillä. (Amazon Web Services, Inc.)

Amazon SQS mahdollistaa mikropalveluiden integroinnin ilman, että mikropalvelut olisi sidottava toisiinsa muiden ohjelmistokomponenttien avulla, mikä mahdollistaa mikropalveluiden pitämisen erillään. SQS on täysin skaalautuva palvelu, joten mikropalveluiden skaalautuessa myös niiden välinen SQS:n avulla toteutettu integraatio skaalautuu. Amazon SQS pystyy siirtämään dataa suurella suorituskyvyllä datan määrästä riippumatta. (Amazon Web Services, Inc.)

Hajautettu viestintäjärjestelmä koostuu kolmesta osasta: mikropalveluiden komponenteista, viestijonosta ja viestijonossa siirrettävistä viesteistä. Viestin tuottaja, eli mikropalvelukomponentti, lähettää viestin jonoon, jonka jälkeen viesti siirretään Amazon SQS-palvelimelle. Viesti pysyy jonossa, kunnes kuluttaja, eli toinen mikropalvelukomponentti, on valmiina käsittelemään viestiä. Kun kuluttaja on käsitellyt viestin, se poistaa viestin jonosta, jotta viestiä ei voida vastaanottaa ja käsitellä uudestaan. (Amazon Web Services, Inc.)

Amazon SQS tarjoaa kahdentyyppisiä viestijonoja. Vakioviestijonoilla on suurin suorituskyky ja rajaton määrä API-kutsuja. Vakioviestijono takaa, että viesti toimitetaan vähintään kerran. Vakioviestijonossa viestit pyritään toimittamaan parhaassa mahdollisessa järjestyksessä, mutta joissakin tapauksissa viestien järjestys saattaa muuttua. Kuvassa 9. havainnollistetaan vakioviestijonon toimintaa, jossa viestit toimitetaan mahdollisimman tehokkaasti. (Amazon Web Services, Inc.)



Kuva 9. Amazon SQS vakioviestijono (Amazon Web Services, Inc.)

Toinen SQS-viestijonotyyppi on FIFO (First In First Out) -viestijono. FIFO-viestijonossa viestit toimitetaan siinä järjestyksessä, jossa ne on asetettu jonoon. Kuva 10. kuvaa FIFO-viestijonon toimintaa. FIFO-viestijonot toimittavat viestin ainoastaan yhden kerran. FIFO-viestijonot pystyvät toimittamaan enintään kolme tuhatta viestiä sekunnissa ja suorittamaan 300 API-kutsua sekunnissa. (Amazon Web Services, Inc.)

Kuva 10. Amazon SQS FIFO-viestijono (Amazon Web Services, Inc.)

SQS-viestejä voidaan käyttää AWS Lambda-funktioiden tapahtumalaukaisimina. Tällöin SQS-viestit sisältävät tiedot, jotka Lambda-funktio käsittelee. AWS Lambda pystyy myös hakemaan viestejä jonosta, kun funktioita suoritetaan. Samaa SQS-viestijonoa pystytään käyttämään useiden eri Lambda-funktioiden kanssa. (Amazon Web Services, Inc.)

Amazon Simple Notification Service (SNS) on viestinvälityspalvelu, joka toimittaa viestejä julkaisijoilta tilaajille. Julkaisijat kommunikoivat asynkronisesti tilaajien kanssa lähettämällä viestejä viestiketjuun, jota tilaajat kuuntelevat. Tilaajat voidaan asettaa kuuntelemaan SNS-viestiketjua, ja vastaanottamaan viestiketjuun julkaistuja viestejä päätepisteiden, kuten esimerkiksi Amazon SQS:n tai AWS Lambdan, avulla. (Amazon Web Services, Inc.)

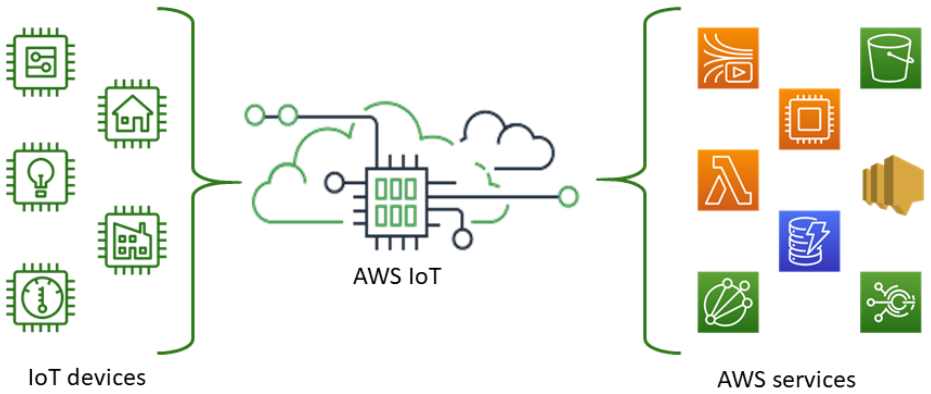
Amazon SNS voi toimia esimerkiksi mikropalveluiden välisenä viestinvälittäjänä. SNS-viestiketjujen tilaajina voidaan käyttää erilaisia AWS:n palveluja. Amazon SNS-viestiketjujen tilaajiksi voidaan määrittää esimerkiksi AWS-Lambda-funktioita,

Amazon SQS-viestijonoja tai HTTP- tai HTTPS-päätepisteitä. (Amazon Web Services, Inc.)

Amazon SNS tukee vakio- ja FIFO-viestiketjuja. Vakioviestiketjua käytettäessä viestien toimitusjärjestys ei ole taattu. Sen sijaan FIFO-viestiketjussa viestit toimitetaan siinä järjestyksessä, kun ne on asetettu viestiketjuun. (Amazon Web Services, Inc.)

3.3.4 AWS IoT Core

AWS IoT sisältää pilvipalveluja, joiden avulla IoT-laitteet voidaan yhdistää AWS-pilvilaskenta-alustaan. AWS IoT mahdollistaa IoT-laitteiden integroinnin pilvipalvelussa toimiviin ohjelmistoratkaisuihin. AWS IoT Core on viestinvälittäjänä IoT-laitteiden ja AWS-pilvilaskenta-alustan välillä. AWS IoT:n roolia pilvilaskenta-alustan ja IoT-laitteen välillä havainnollistetaan kuvassa 11. (Amazon Web Services, Inc.)



Kuva 11. AWS IoT (Amazon Web Services, Inc.)

AWS IoT:n avulla ohjelmistoratkaisuja voidaan toteuttaa hyödyntämällä erilaisia IoT-laitteita. AWS IoT:n avulla ohjelmistot pystyvät saamaan tietoa laitteilta, kuten erilaisilta sensoreilta tai käyttöliittymiltä. IoT-laitteiden lähettämää tietoa voidaan käsitellä AWS:n palveluiden, kuten AWS Lambdan avulla. (Amazon Web Services, Inc.)

AWS IoT Core tukee useita tiedonsiirtoprotokollia, kuten MQTT (Message Queuing and Telemetry Transport), MQTT WSS:n avulla (Websockets Secure), HTTPS (Hypertext Transfer Protocol - Secure), ja LoRaWAN (Long Range Wide Area Network). IoT-laitteet yhdistetään IoT Core:en päätepisteiden avulla. Päätepisteet

mahdollistavat toimintoja, jotka ohjaavat ja hallitsevat IoT-ratkaisua. Päätepisteet ovat aluekohtaisia. (Amazon Web Services, Inc.)

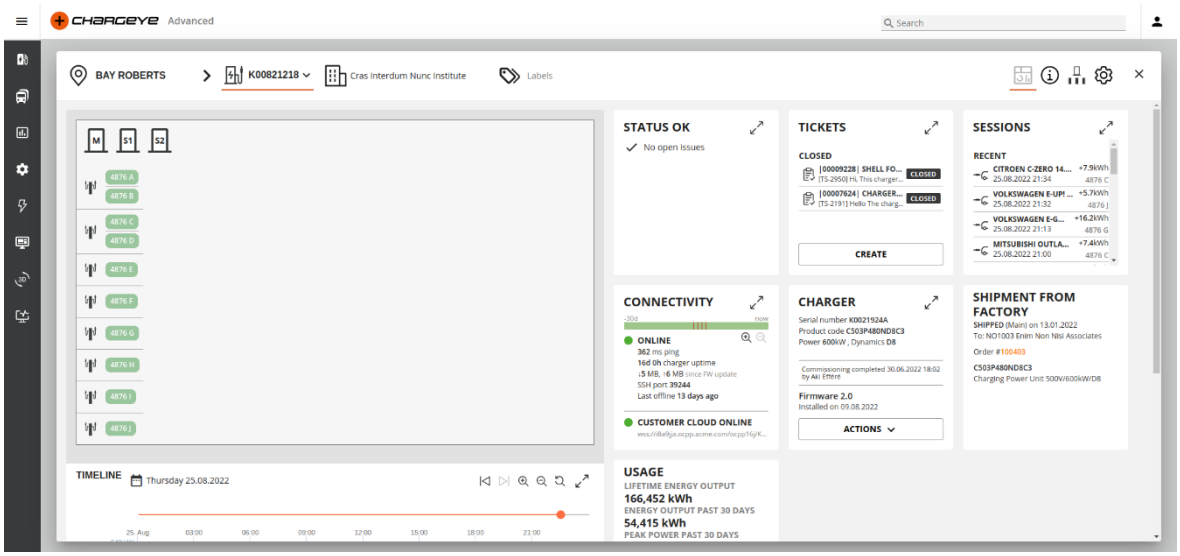
MQTT on kevyt ja paljon käytetty viestintäprotokolla. MQTT on suunniteltu tiedonsiirroltaan rajoittuneille laitteille. AWS IoT tukee MQTT:tä tietyillä eroavaisuuksilla. IoT-laitteet julkaisevat otsikoiden perusteella tunnistettavia MQTT-viestejä AWS IoT:lle. Vastaavasti AWS IoT julkaisee MQTT-viestejä ilmoittaakseen laitteille ja ohjelmille tapahtumista ja muutoksista. MQTT-viestejä pystyy katsomaan AWS IoT Coren MQTT-asiakkaan avulla. MQTT-asiakas pystyy kuuntelemaan esimerkiksi tietyllä otsikolla julkaistuja viestejä. (Amazon Web Services, Inc.)

4 Dynaaminen kuormanhallinta

4.1 ChargEye:n dynaamiset tehoryhmät

ChargEye on Kempowerin käyttäjälisenssipohjainen SaaS-palvelu, jonka avulla voidaan valvoa ja ohjata latauslaitteita. Se koostuu latauslaitteille tarkoitetusta taustajärjestelmästä ja verkkoselaimella käytettävästä käyttöliittymästä. ChargEye:n ja latauslaitteiden välinen tiedonsiirto perustuu OCPP-standardiin. ChargEye:n ohjelmistoarkkitehtuuri perustuu Amazon Web Services pilvilaskenta-alustalle toteutettuihin mikropalveluihin. ChargEye pohjautuu alun alkaen Kemppi Oy:n WeldEye-järjestelmään, ja se on kehittynyt ainutlaatuiseksi ohjelmistoratkaisuksi Kempowerin toimesta.

Kuvassa 12 on kuvankaappaus ChargEye:n käyttöliittymästä, jossa näkyy Kempowerin valmistaman S-series latausjärjestelmän päänäkymä. Päänäkymässä pystytään näkemään erilaisia tietoja latauslaiteesta, kuten tämänhetkinen tila, viimeisimmät lataustapahtumat ja mahdolliset virheilmoitukset. Päänäkymästä latauslaitteelle voidaan myös lähettää erilaisia komentoja, kuten lataustehon rajoituksia.



Kuva 12. ChargEye:n käyttöliittymä (Kempower)

Kuvassa 12. oleva S-series latausjärjestelmä koostuu kolmen tehomoduulikaapin latausteholyksiköstä ja kahdeksasta satelliitista, jotka muodostavat latauskentän. Tehomoduulikaapit sisältävät tehomoduuleja, jotka tuottavat suuritehoista tasasähkövirtaa. S-series latausjärjestelmän satelliitit toimivat latauspisteinä. ChargEye mahdollistaa kaikkien latausjärjestelmään kuuluvien laitteiden etähallinnan. ChargEye:sta pystyy näkemään esimerkiksi yksittäisen satelliitin lataustapahtumat tai yksittäisen tehomoduulin senhetkisen tilan.

ChargEye sisältää paljon erilaisia toiminnallisuuksia. Sen avulla pystytään esimerkiksi hallitsemaan ja valvomaan latauslaitteita, todentamaan latauslaitteiden käyttäjiä eri menetelmillä, sekä tunnistamaan erilaisia latauslaitteisiin kytkettyjä ajoneuvoja. ChargEye mahdollistaa myös latauspisteoperaattoreiden käyttää omia taustajärjestelmiään yhdessä Kempowerin latauslaitteiden kanssa.

Yksi ChargEyen ominaisuuksista on ”power groups” eli tehoryhmät. Tehoryhmät ovat ominaisuus, joka jakaa tietyn määrän sähkövirtaa ryhmässä olevien latauslaitteiden kesken. Käytännössä tehoryhmien toiminta perustuu lataustehon rajoituksia laskevaan algoritmiin, ja tehoryhmän latauslaitteille asettamiin latausprofiileihin, jotka sisältävät kyseiset rajoitukset.

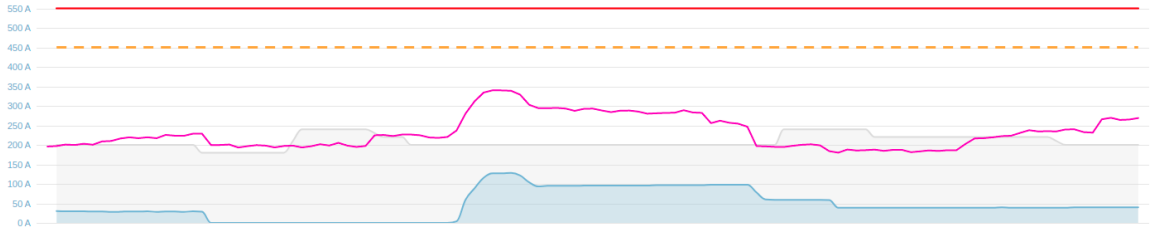
Tehoryhmän tärkein konfiguroitava arvo on kyseiselle ryhmälle määritetty maksimiteho kilowatteina tai sähkövirta ampeereina. Tehoryhmässä olevien latauslaitteiden yhteenlaskettu sähkönkulutus ei koskaan voi ylittää tätä arvoa. Tehoryhmässä saatavilla oleva sähkövirta jaetaan latauslaitteille demokraattisesti sen hetkisen kysynnän ja tarjonnan mukaan. Latausteho jaetaan tasavertaisesti sitä tarvitseville latauslaitteille, mutta latauslaitteita voidaan myös priorisoida tehoryhmän sisällä, jolloin korkeammalla prioriteetilla olevat latauslaitteet saavat ensisijaisesti enemmän sähköenergiaa käyttöönsä.

Dynaaminen kuormanhallinta toteutetaan osaksi ChargEyen power groups -ominaisuutta. Dynaamisen kuormanhallinta-algoritmin ollessa aktivoituna tehoryhmälle määritettyä maksimisähkövirran suuruutta säädetään kiinteistön kokonaissähkönkulutuksen mukaan. Dynaamisen kuormanhallinnan tärkein tehtävä on pitää kiinteistön kokonaissähkönkulutus kiinteistön sähköjärjestelmän sulakkeiden maksimikapasiteetin alapuolella, mutta samalla maksimoida latauslaitteiden latausteho.

Kiinteistön kokonaissähkönkulutuksen noustessa, muiden kuin latauslaitteiden vaikutuksesta, tehoryhmälle määritetyn suurimman sallitun sähkövirran määrää pienennetään, jolloin latauslaitteiden suurin sallittu yhteenlaskettu kokonaiskulutus laskee. Vastaavasti, jos kiinteistön kokonaissähkönkulutus muiden sähkölaitteiden toimesta pienenee, niin tehoryhmän suurimman sallitun sähkövirran määrää nostetaan. Näin pystytään varmistamaan, että latauslaitteilla on aina käytössä niin paljon sähkövirtaa kuin kiinteistön sähköliittymällä on niille tarjota.

Tehoryhmiä voidaan monitoroida ja hallita ChargEye:n käyttöliittymän avulla. Dynaamista kuormanhallinta-algoritmia käyttävät tehoryhmät sisältävät useita käyttäjän määrittämiä parametreja, jotka voidaan asettaa käyttöliittymästä. Kuvassa 13 on kuvankaappaus tehoryhmien käyttöliittymästä, joka sisältää dynaamisen kuormanhallinta-algoritmin käyttämien

tietojen muodostaman kuvaajan. Violetti viiva kuvaa kiinteistön kokonaissähkönkulutusta ja sininen alue tehoryhmän latauslaitteiden kokonaissähkönkulutusta.



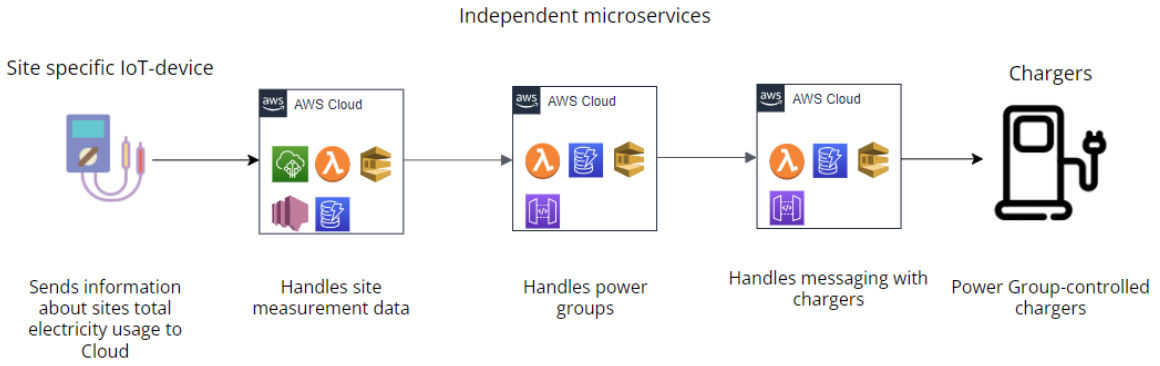
Kuva 13. Dynaamisen kuormanhallinnan toimintakuvaaja (Kempower)

4.2 Palveluarkkitehtuuri

Dynaamisen kuormanhallinnan ohjelmisto perustuu palvelimettoman tiedonsiirtomallin mukaiseen arkkitehtuuriin. Ohjelmisto koostuu useista eri mikropalveluista, jotka toimivat AWS-pilvilaskenta-alustalla. Ohjelmistossa hyödynnetään useita AWS:n tarjoamia palveluja, kuten AWS Lambda:a, AWS DynamoDB:tä, AWS IoT Core:a ja AWS SQS:ää. Ohjelmiston toteuttamisessa käytettävät ohjelmointiteknologiat ovat TypeScript, Node.js ja Serverless Framework. Ohjelmiston toimintaan liittyy myös olennaisesti kiinteistön kokonaissähkönkulutuksesta ajankohtaista tietoa ohjelmistolle lähettävä IoT-laite, ja ohjelmiston avulla ohjattavat latauslaitteet.

IoT-laite, joka lähettää ajankohtaista tietoa kiinteistön sen hetkisestä kokonaissähkönkulutuksesta ohjelmistolle, on sijoitettu kiinteistön sähköjärjestelmän pääkeskuksen läheisyyteen. Laite lukee kiinteistön kokonaissähkönkulutusta mittaavalta laitteelta tiedot, jotka sisältävät senhetkisen sähkövirran suuruuden jokaiselle kolmelle vaiheelle. IoT-laite lähettää tiedot AWS IoT Core-palveluun MQTT-viestien avulla.

AWS IoT Core-palvelussa kiinteistön kokonaissähkönkulutuksen sisältävät tiedot lähetetään tiedonkäsittelykanavana toimivalle mikropalvelulle, jonka avulla tiedot saadaan liitettyä tietyn kiinteistön latauslaitteet sisältävän tehoryhmän tietokantaobjektiin. Toinen mikropalvelu hallitsee tehoryhmiä. Mikropalvelussa algoritmit määrittävät maksimisähkövirran suuruuden tehoryhmälle ja lataustehon rajoitukset jokaiselle ryhmässä olevalle latauslaitteelle. Lataustehon rajoitukset lähetetään erilliselle mikropalvelulle, joka sisällyttää tehorajoitukset latausprofiileihin ja lähettää ne latauslaitteille. Kuvassa 14 on kuvattu ohjelmiston palveluarkkitehtuuria.



Kuva 14. Dynaamisen kuormanhallintaohjelmiston palveluarkkitehtuuri

Tehoryhmän ohjaamat latauslaitteet on sijoitettu samaan kiinteistöön, jonka kokonaissähkönkulutuksesta kertovia tietoja käytetään tehoryhmän ohjaukseen. Dynaamisen kuormanhallintaohjelmiston tavoitteena on vaikuttaa kiinteistön kokonaissähkönkulutukseen asettamalla latauslaitteille lataustehoa rajoittavia latausprofiileja. Tehoryhmän toiminnan kannalta oleellista on, että kaikki tehoryhmässä olevat latauslaitteet käyttävät samaa sähköliittymää.

4.3 Mikropalvelut

Dynaaminen kuormanhallinta on toteutettu AWS-pilvilaskenta-alustalle erilaisten mikropalveluiden avulla. Mikropalvelut vastaavat niille määritetyistä tehtävistä. Ohjelmiston toiminnallisuus muodostuu kaikkien siihen kuuluvien mikropalveluiden toiminnallisesta kokonaisuudesta, jossa jokaisella mikropalvelulla on oma tehtävänsä. Mikropalveluiden toiminta on toisistaan riippumatonta, mutta dynaaminen kuormanhallinta tarvitsee kaikki siihen liittyvät mikropalvelut toimiakseen.

Yksi kuormanhallintaohjelmiston mikropalvelu vastaa IoT-laitteen lähettämien tietojen käsittelystä. Se sisältää tiedonkäsittelykanavan IoT-laitteen kiinteistön sähkönkulutuksesta lähettämälle tiedolle. Mikropalvelu sisältää AWS IoT Core resurssien määritykset MQTT-viestien käsittelyä varten, tietokantataulukon kiinteistön kokonaissähkönkulutuksen tietojen tallentamiseen, sekä tietojen käsittelyä hoitavat Lambda-funktiot. Funktiot tarkastavat tietojen oikeellisuuden, ja tallentavat tiedot niitä koskevien tehoryhmien tietokantaobjekteihin. IoT-laitteen lähettämää tietoa käsittelevän mikropalvelun rakennetta ja toimintaa kuvaillaan yksityiskohtaisemmin tämän dokumentin kappaleessa 4.4.

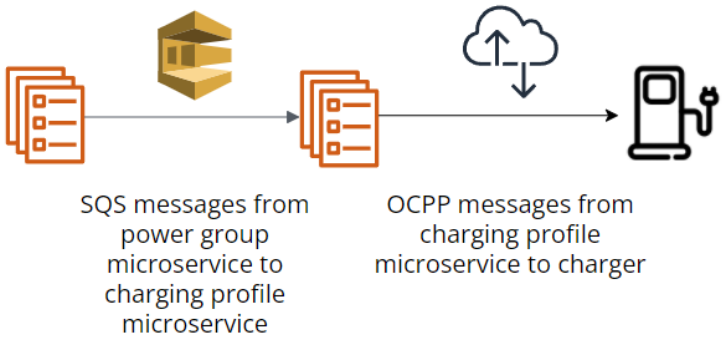
Dynaamisen kuormanhallinnan toiminta perustuu niin sanottuihin toimintakierroksiin. Yhden toimintakierroksen aikana mikropalvelut käsittelevät tehoryhmiin liittyviä tietoja, esimerkiksi algoritmien avulla. Algoritmit on sisällytetty Lambda-funktioihin, jotka ajetaan kerran jokaista

tehoryhmää kohden jokaisella toimintakierroksella. Toimintakierrokset on ajastettu askelfunktion avulla, joka suoritetaan tietyin väliajoin.

Tehoryhmiä hallitseva mikropalvelu koostuu kahdesta osasta. Mikropalvelu sisältää ohjelmiston toiminnallisuuden, joka ohjaa tehoryhmiä. Mikropalvelu sisältää useita Lambda-funktioita, jotka käsittelevät tehoryhmiin liittyviä tietoja. Mikropalvelun sisäinen viestintä on toteutettu SQS-viestien avulla, jotka toimivat tapahtumalaukaisimina Lambda-funktioille. Nämä Lambda-funktiot sisältävät esimerkiksi yksittäisten latauslaitteiden tehorajoituksia laskevan algoritmin sekä dynaamisen kuormanhallinnan algoritmin, joka laskee tehoryhmän suurimman sallitun sähkövirran määrän. Dynaamista kuormanhallintaa toteuttavan algoritmin toimintaa ja sen käyttämiä lähtötietoja on kuvattu yksityiskohtaisemmin tämän dokumentin kappaleessa 4.5.

Tehoryhmiä hallitseva mikropalvelu sisältää tehoryhmien toiminnallisuutta ohjaavan algoritmin lisäksi rajapintoja, joiden avulla voidaan hallita tehoryhmiä. Rajapintojen avulla pystytään hakemaan tehoryhmien tiedot, määrittämään tehoryhmän konfiguraatio, luomaan ja poistamaan tehoryhmä, sekä lisäämään latauslaiteita tehoryhmään. Rajapinnat on sidottu eri Lambda-funktioihin, jotka sisältävät pyyntöjä käsittelevät loogiset ohjelmat. Rajapintoja voidaan kutsua ChargEye:n käyttöliittymän avulla, mikä edellyttää käyttäjän tunnistautumista.

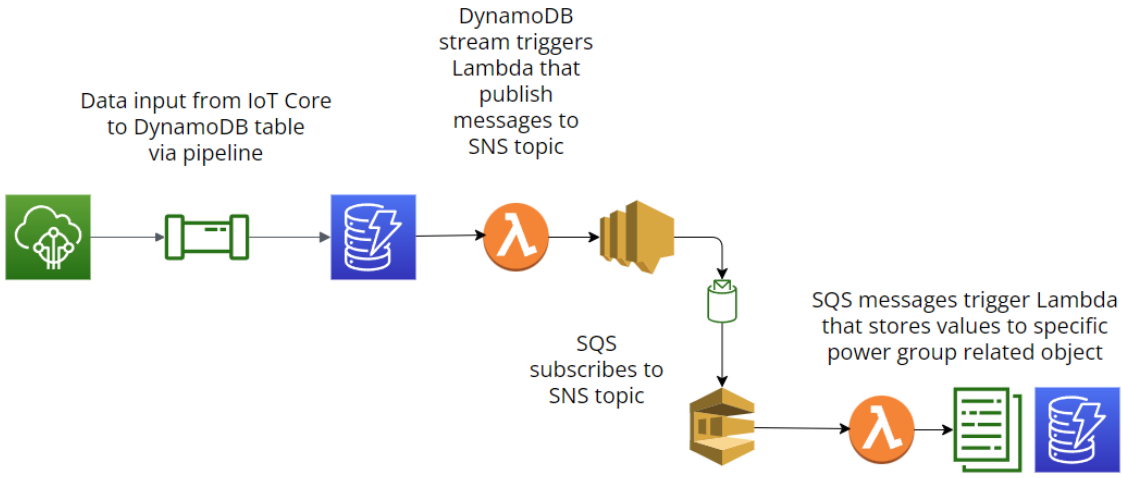
Tehorajoitukset sisältävät latausprofiilit lähetetään latauslaitteille itsenäisen mikropalvelun avulla. Latauslaitteille lasketut tehorajoitukset lähetetään tehoryhmiä hallitsevasta mikropalvelusta SQS-viesteinä, jotka toimivat viestejä käsittelevän Lambda-funktion tapahtumalaukaisimina. Funktio luo latausprofiilit, jotka sisältävät tehorajoitukset. Funktio lähettää latausprofiilit rajapinnalle, jonka avulla profiilit lähetetään OCPP-standardin mukaisella tiedonsiirrolla latauslaitteille. Kuvassa 15 havainnollistetaan latausprofiileja latauslaitteelle lähettävän mikropalvelun tiedonsiirtoa.



Kuva 15. Latausprofiileja käsittelevän mikropalvelun tiedonsiirto

4.4 IoT-laitteen lähettämän datan käsittely

Kiinteistön kokonaissähkönkulutuksesta kertovan tiedon käsittelyä hoitaa itsenäinen mikropalvelu. Mikropalvelun tehtävä on käsitellä IoT-laitteelta lähetetyt tiedot niin, että ne tallennetaan oikeaan tehoryhmään liittyvään tietokantataulukossa sijaitsevaan objektiin. Mikropalvelun arkkitehtuuria on kuvattu kuvassa 16.



Kuva 16. IoT-laitteen lähettämää tietoa käsittelevä mikropalvelu

AWS IoT Core-palvelu sisältää rajapinnan, johon IoT-laite lähettää kiinteistön kokonaissähkönkulutuksesta kertovat MQTT-viestien avulla. Laitteelle on määritetty käyttöoikeudet IoT Core-palveluun ja kyseiset käyttöoikeudet on sidottu tiettyyn sertifikaattiin. IoT Core tarkastaa laitteen käyttämän sertifikaatin ja myöntää laitteelle oikeuden lähettää MQTT-viestejä palvelun rajapinnalle.

Viestien sisältämät tiedot tallennetaan objekteina DynamoDB:n tietokantataulukkoon. Koska IoT-laite lähettää kiinteistön sähköjärjestelmän kolmen vaiheen tiedot omina viesteinään, oleellista on tallentaa kunkin tiedon viestit oikeaan objektiin. IoT Core:ssa on määritelty attribuutit, joiden perusteella MQTT-viestien sisältö tallennetaan tietokantataulukon objekteihin. IoT-laitteen sijainti määrittää objektien partitioavaimen ja objektien lajitteluavain määräytyy viestien aikaleimojen mukaan. Saman aikaleiman sisältävät viestit tallennetaan samaan objektiin.

Tietokantataulukkoon on luotu tapahtumalaukaisin, jonka avulla kutsutaan Lambda-funktiota. Tapahtumalaukaisin aktivoituu, kun tietokantataulukkoon lisätään tietoa. Lambda-funktiolle lähetetään lisätyn tai muokatun objektin tiedot. Funktio määrittelee, sisältääkö

objekti kaikki tarvittavat tiedot, jotka tarvitaan tehoryhmiä ohjaavassa algoritmissa. Jos objekti sisältää kaikki tarvittavat tiedot, nämä tiedot julkaistaan Simple Notification Service -palvelun viestiketjuun (SNS Topic). Viestiketjua kuuntelee SQS-jono, joka poimii viestit ketjusta.

SQS-jonon viestit toimivat tapahtumalaukaisimena mikropalvelun viimeiselle Lambda-funktiolle. Funktion tehtävä on tarkastaa viestien sisältö ja päivittää viimeisimmät tiedot kiinteistön kokonaissähkönkulutuksesta tiettyä tehoryhmää koskevaan objektiin tietokantataulukossa. Tiedoista tarkastetaan viestien aikaleima, jonka avulla määritetään tiedon luotettavuus. Tiedot tallennetaan vain, jos niiden aikaleima on tarpeeksi tuore, ja tiedot kattavat sähköjärjestelmän jokaisen kolmen vaiheen.

IoT-laitteen lähettämiä tietoja käsittelevä mikropalvelu on olennainen osa dynaamisen kuormanhallinnan toimintaa. Tiedot on käsiteltävä riittävän nopeasti, jotta tehoryhmät pystyvät vaikuttamaan kiinteistön kokonaissähkönkulutukseen ennen kuin kiinteistönsähköliittymän maksimikapasiteetti saavutetaan. Tietojen käsittelyssä olennaista on myös tietojen oikeellisuuden tarkastaminen. Tehoryhmien on saatava ajankohtaista tietoa, jotta ne pystyvät vaikuttamaan kiinteistön kokonaissähkönkulutukseen toivotulla tavalla.

4.5 Dynaaminen kuormanhallinta-algoritmi

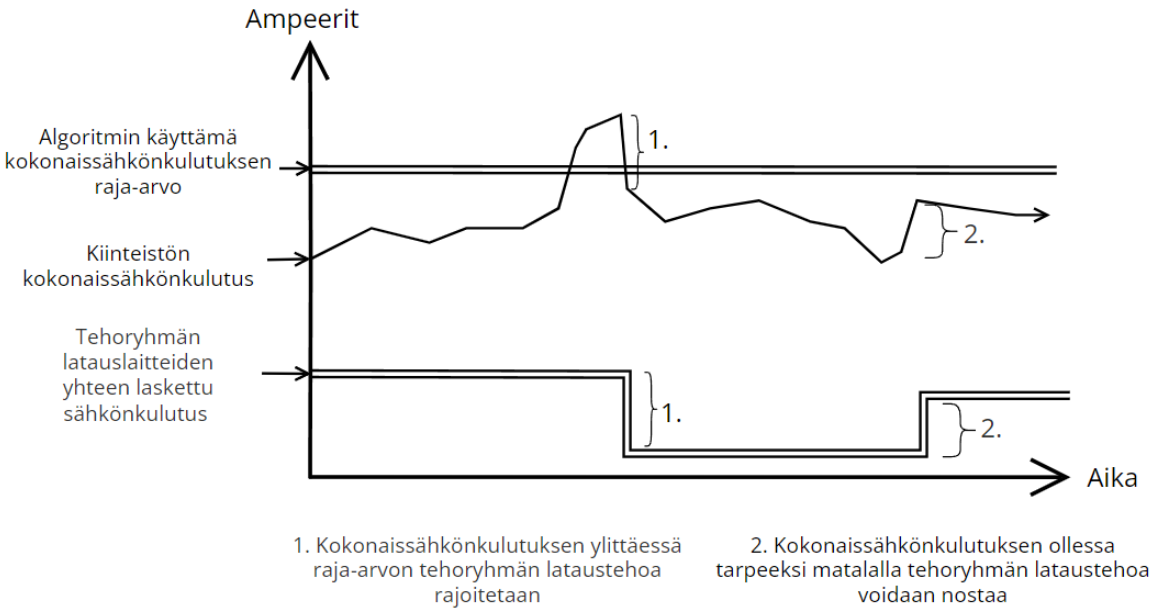
Tehoryhmille, jotka ohjaavat tiettyä joukkoa latauslaitteita, on määritetty suurin sallittu sähkövirran määrä. Tehoryhmässä olevien latauslaitteiden yhteenlaskettu sähkönkulutus ei koskaan voi ylittää tätä arvoa. Ryhmässä oleville latauslaitteille jaetaan saatavilla olevaa sähkövirtaa lataustehon kysynnän mukaan varaamalla tietty määrä sähkövirtaa aina tietyn latauslaitteen käyttöön. Ryhmässä olevien latauslaitteiden sähkönkulutusta ohjataan asettamalla niille lataustehoa rajoittavia latausprofiileja tehoryhmän kyseiselle latauslaitteelle varaaman sähkövirran mukaisesti.

Dynaaminen kuormanhallinta-algoritmi säätää tietyn tehoryhmän suurinta sallittua sähkövirran määrää. Dynaamista kuormanhallinta-algoritmia käyttävälle tehoryhmälle on määritetty suurin kiinteistön kokonaissähkönkulutuksen määrä, jota ei koskaan tule ylittää. Tämä arvo voi olla esimerkiksi sähköjärjestelmän sulakekoko. Tehoryhmän käytössä olevan sähkövirran maksimikapasiteetti tulee määrittää latauslaitteiden käyttöympäristön mukaan yksilöllisesti. Kiinteistön sähkövirran maksimikapasiteetin lisäksi tehoryhmälle on määritetty kiinteistön kokonaissähkönkulutuksen raja-arvo, johon senhetkistä kiinteistön kokonaissähkönkulutusta ja tehoryhmän latauslaitteiden yhteenlaskettua sähkönkulutusta verrataan.

Algoritmi toimii sille määritettyjen sääntöjen mukaisesti. Algoritmi on toteutettu TypeScript:llä ehtolauseiden ja loogisten operaattoreiden avulla. Ehtolauseilla on määritetty

erilaisia laskentakaavoja ja lopputuloksia, joiden käyttö riippuu tehoryhmän latauslaitteiden ja kiinteistönsähkönkulutuksen senhetkisestä tilasta.

Algoritmin lähtötietoina toimivat kiinteistön kokonaissähkönkulutuksesta kertovat tiedot, latauslaitteiden sähkönkulutuksesta kertovat tiedot, sekä kiinteistön sähköjärjestelmän maksimisähkövirran perusteella määritetty raja-arvo. Tehoryhmän latauslaitteiden käytössä oleva maksimisähkövirran suuruus määritetään vertaamalla senhetkistä kiinteistön kokonaissähkönkulutusta tehoryhmälle määritettyyn kokonaissähkönkulutuksen raja-arvoon. Tehoryhmä rajoittaa latauslaitteiden lataustehoa niin, että kiinteistön kokonaissähkönkulutus pysyy raja-arvon alapuolella. Raja-arvon määrittämä kokonaissähkönkulutus voidaan hetkellisesti ylittää, minkä vuoksi raja-arvo on todellista kiinteistön sähköjärjestelmän ehdotonta maksimikapasiteettia pienempi. Algoritmin toimintaperiaatetta havainnollistetaan yksinkertaistetusti kuvassa 17.



Kuva 17. Dynaamisen kuormanhallinta-algoritmin toimintaperiaate

Kiinteistön kokonaissähkönkulutuksen noustessa muiden sähköä kuluttavien laitteiden toimesta, latauslaitteiden käytössä olevaa lataustehoa pienennetään niin, että kokonaissähkönkulutus pysyy raja-arvon alapuolella. Vastaavasti, jos kiinteistön kokonaissähkönkulutus on huomattavan paljon raja-arvon alapuolella, niin tehoryhmän latauslaitteiden yhteenlaskettua sähkövirran määrää voidaan nostaa.

Algoritmi pyrkii reagoimaan kiinteistön kokonaissähkönkulutuksen heittelyihin ennen raja-arvon ylittymistä. Koska algoritmi perustuu reaaliaikaiseen tietoon kiinteistön

sähkönkulutuksesta, algoritmin on mahdotonta reagoida kokonaissähkönkulutuksen äkillisiin nousuihin ennakoivasti. Joskus kiinteistön kokonaissähkönkulutus saattaa nousta hetkellisesti niin nopeasti, että raja-arvo ylitetään. Tämän vuoksi raja-arvo määritetään riittävällä marginaalilla kiinteistön sähköjärjestelmän maksimikapasiteettia pienemmäksi. Riittävä marginaali pystytään määrittämään tarkkailemalla kiinteistön kokonaissähkönkulutusta tietyn ajanjakson aikana.

Algoritmin toiminnan kannalta oleellista on, että tiedot kiinteistön kokonaissähkönkulutuksesta ovat ajankohtaisia. Tietojen aikaleima tarkastetaan aina, kun tietoja käytetään tehoryhmän ohjaukseen. Mikäli tietojen aikaleima on liian vanha, tehoryhmä asetetaan viansietotilaan. Viansietotilaa varten tehoryhmälle on asetettu maksimisähkövirran määrä, jota tehoryhmä käyttää ollessaan viansietotilassa. Näin varmistetaan, että tehoryhmässä olevat latauslaitteet eivät ylikuormita kiinteistön sähköjärjestelmää esimerkiksi IoT-laitteen verkkoyhteyden katketessa. Tehoryhmä palautuu viansietotilasta normaaliin tilaan heti, kun ajankohtaista tietoa kiinteistön kokonaissähkönkulutuksesta on saatavilla.

Kun dynaaminen kuormanhallinta-algoritmi on laskenut suurimman sallitun sähkövirran määrän tehoryhmälle, latauslaitteiden tehorajoituksia laskeva algoritmi laskee uudet lataustehon rajoitukset tehoryhmässä oleville latauslaitteille. Latauslaitteiden senhetkinen lataustehon kysyntä määrittää rajoitukset jokaiselle latauslaitteelle yksilöllisesti. Jokaiselle tehoryhmässä olevalle latauslaitteelle varataan algoritmin laskema määrä sähkövirtaa, jonka yhteenlaskettu määrä ei koskaan voi ylittää dynaamisen kuormanhallinta-algoritmin määrittämää tehoryhmän suurinta sallittua kokonaissähkövirtaa.

Lataustehon rajoituksia laskeva algoritmi jakaa tehoryhmässä saatavilla olevan sähkövirran latauslaitteille. Algoritmi määrittää yksilölliset tehorajoitukset latauslaitteille niiden senhetkisen lataustehonkysynnän mukaan. Jokaisen latauslaitteen suorittaessa lataustapahtumaa latausteho jaetaan demokraattisesti kaikkien latauslaitteiden kesken. Jos tehoryhmässä vain yhdellä latauslaitteella on sillä hetkellä aktiivinen lataustapahtuma, tämä latauslaite saa käyttöönsä niin paljon lataustehoa kuin tehoryhmässä on saatavilla. Poikkeuksena voidaan pitää tilannetta, jossa latauslaitteita on priorisoitu, jolloin lataustehoa jaetaan ensisijaisesti niille latauslaitteille, jotka on priorisoitu korkeammalle.

Kun algoritmi on määrittänyt lataustehon rajoitukset jokaiselle tehoryhmässä olevalle latauslaitteelle, ne lähetetään erilliselle mikropalvelulle SQS-viestien avulla. SQS-viestit sisältävät lataustehon rajoitukset tehoryhmässä oleville latauslaitteille. Tehoryhmien laskemat lataustehon rajoitukset muodostetaan OCPP-standardin mukaisiksi Smart Charging-profiileiksi Lambda-funktion avulla. Lambda-funktio lähettää latausprofiilit rajapinnalle, jonka avulla ne asetetaan latauslaitteille.

5 Yhteenveto

Työn päämäärä oli luoda automatisoitu suurteholatureiden hallintajärjestelmä, jonka avulla pystytään hyödyntämään kiinteistön varaamaton sähkövirran kapasiteetti mahdollisimman tehokkaasti latauslaiteiden käyttöön. Lopputuloksena syntyi dynaaminen kuormanhallinta-ohjelmisto, joka pystyy säätämään latauslaitteiden lataustehoa kiinteistön kokonaissähkönkulutuksen mukaan. Työ toteutettiin osaksi Kempowerin ChargEye:n ”power groups” -ominaisuutta, jonka avulla voidaan määrittää yhteinen sähkövirran käytön rajoitus ryhmässä oleville latauslaitteille, ja rajoittamaan ryhmän latauslaitteiden lataustehoa automatisoidusti.

Dynaaminen kuormanhallintaohjelmisto on otettu pilottikäyttöön kesäkuussa 2022. Ohjelmiston toimintaa on monitoroitu käyttöönotosta lähtien. Ohjelmisto on mahdollistanut latauslaitteiden toiminnan mahdollisimman suurella latausteholla kiinteistön sähköliittymän ja muiden laitteiden tuottaman sähkövirrankäytön rajoitusten puitteissa.

Dynaamista kuormanhallintaa pystytään hyödyntämään kohteissa, joissa halutaan maksimoida latauslaitteiden tarjoama latausteho, mutta kiinteistön sähköliittymä tai muut tekijät rajoittavat kiinteistön kokonaissähkönkulutusta. Dynaaminen kuormanhallinta toimii ratkaisuna kohteisiin, joissa kokonaissähkönkulutus on vaihtelevaa. Sähköautojen lisääntyessä myös latauslaitteiden erilaiset käyttöympäristöt lisääntyvät. Dynaaminen kuormanhallinta pystyy joissakin tilanteissa tarjoamaan joustavuutta latauslaitteiden hankintaan ilman, että kiinteistön sähköliittymää joudutaan muuttamaan radikaalisti.

Työssä toteutettiin ohjelmistoratkaisu palvelimettoman tiedonkäsittelymallin mukaisesti. Palvelimeton tiedonkäsittelymalli mahdollistaa ohjelmiston skaalautuvuuden tulevaisuudessa. Ohjelmisto toteutettiin Amazon Web Services -pilvilaskenta-alustalle Serverless -ohjelmointikehyksen avulla. Ohjelmiston kehittämiseen käytetyt teknologiat ovat suosittuja, minkä uskotaan parantavan ohjelmiston ylläpitomahdollisuuksia samojen teknologioiden avulla jatkossa.

Toteutunut dynaaminen kuormanhallinta-algoritmi luo pohjan ominaisuuden jatkokehitykselle. Algoritmin toimintaa voitaisiin kehittää esimerkiksi kiinteistön kokonaissähkönkulutusta ennakoivaksi esimerkiksi koneoppimismallien avulla. Ohjelmiston ylläpito ja mahdollinen jatkokehitys tapahtuu Kempowerin tuotekehitysyksikön toimesta.

Lähteet

Amazon Web Services, Inc. Amazon Simple Queue Service. Viitattu: 30.8.2022. Saatavissa: https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html

Amazon Web Services, Inc. AWS IoT Core. Viitattu: 30.8.2022. Saatavissa: https://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html

Amazon Web Services, Inc. AWS DynamoDB. Viitattu: 26.8.2022. Saatavissa: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.CoreComponents.html

Amazon Web Services, Inc. AWS Lambda. Viitattu: 22.8.2022. Saatavissa: https://docs.aws.amazon.com/lambda/latest/dg/welcome.html

Amazon Web Services, Inc. 2022. What is AWS. Viitattu: 21.8.2022. Saatavissa: https://aws.amazon.com/what-is-aws/

Amazon Web Services, Inc. What is cloud computing? Viitattu: 22.8.2022. Saatavissa: https://aws.amazon.com/what-is-cloud-computing

Autoalan tiedotuskeskus. 2022. Ensirekisteröityjen henkilöautojen käyttövoimatilastot. Viitattu 4.9.2022. Saatavissa: https://www.aut.fi/tilastot/ensirekisteroinnit/ensirekisteroinnit_kayttovoimittain/henkiloautojen_kayttovoimatilastot

Blink Charging Co. 2022. Networked vs. Non-Networked Chargers for Hosts. Viitattu: 18.8. Saatavissa: https://blinkcharging.com/understanding-networked-vs-non-networked-chargers-for-host-locations/

Cloudflare. What is serverless computing? Viitattu: 21.8.2022. Saatavissa: https://www.cloudflare.com/learning/serverless/what-is-serverless/

Elenia. 2022. Sähköinen liikenne. Viitattu 14.8.2022. Saatavissa: https://www.elenia.fi/tulevaisuuden-energia/sahkontuotanto-ja-kulutus/sahkoinen-liikenne

Elenia. 2022. Sähköautoilu. Viitattu 14.8.2022. Saatavissa: https://www.elenia.fi/palvelut/kotiin-ja-mokille/sahkoautoilu

Engdahl, H. 2021. Toimiva latausstrategia laajentaa sähkökuorma-auton kantamaa. Viitattu: 17.8.2022. Saatavissa: https://www.volvotrucks.fi/fi-fi/news/insights/articles/2021/nov/How-a-good-charging-strategy-can-extend-an-electric-trucks-range.html

EVBox. 2022. Everything you should know about electric vehicle charging. Viitattu 11.8.2022. Saatavissa: https://evbox.com/en/ev-charging-guide

Greenlots. 2018. Open vs. Closed Charging Stations: Advantages and Disadvantages Viitattu: 20.8.2022. Saatavissa: https://greenlots.com/wp-content/uploads/2018/09/Open-Standards-White-Paper.pdf

Kempower. Kuvakaappaukset ChargEye:n käyttöliittymästä. Viitattu 28.8.2022.

Liikennevirasto. 2017. Selvitys sähköbussien edistämiseksi suomalaisilla kaupunkiseuduilla. Viitattu 14.8.2022. Saatavissa: https://www.motiva.fi/files/14796/Selvitys_sahkobussien_edistamiseksi_suomalaisilla_kaupunkiseuduilla.pdf

Lintilä, R. 2017. Serverless – mitä se tarkoittaa ja miksi siitä pitäisi kiinnostua? Viitattu: 21.8.2022. Saatavissa: https://www.solita.fi/blogit/serverless-mita-se-tarkoittaa-ja-miksi-siita-pitaisi-kiinnostua/

Microsoft. 2022. TypeScript for JavaScript Programmers. Viitattu: 22.8.2022. Saatavissa: https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html

Microsoft. 2022. TypeScript for the New Programmer. Viitattu: 21.8.2022. Saatavissa: https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html

Node.js. Introduction to Node.js. Viitattu: 21.8.2022. Saatavissa: https://nodejs.dev/en/learn/introduction-to-nodejs

Npm. About npm. Viitattu: 22.8.2022. Saatavissa: https://docs.npmjs.com/about-npm

Open Charge Alliance. OCA - Information Leaflet. Viitattu 20.8.2022. Saatavissa: https://www.openchargealliance.org/uploads/files/OCA-Information_Leaflet.pdf

Open Charge Alliance. Protocols. Viitattu: 18.8.2022. Saatavissa: https://www.openchargealliance.org/protocols/

Open Charge Alliance. 2017. Open Charge Point Protocol 1.6, edition 2. Viitattu 20.8.2022.

Open Charge Alliance. 2020. Open Charge Point Protocol 2.0.1. Viitattu 20.8.2022.

Orcioni, S & Conti, M. 2020. EV Smart Charging with Advance Reservation Extension to the OCPP Standard. Viitattu: 20.8.2022.

ReadWrite. 2013. What You Need To Know About Node.js. Viitattu: 22.8.2022. Saatavissa: https://readwrite.com/what-you-need-to-know-about-nodejs/

Red Hat. 2022. What is serverless? Viitattu: 21.8.2022. Saatavissa: https://www.redhat.com/en/topics/cloud-native-apps/what-is-serverless

Serverless, Inc. Serverless Framework Concepts. Viitattu 22.8.2022. Saatavissa: https://www.serverless.com/framework/docs/providers/aws/guide/intro

Teknologian teollisuus. 2022. Suurteholatausasemien (>100kW) kasvu jatkuu voimakkaana. Viitattu 11.8.2022. Saatavissa: https://emobility.teknologiateollisuus.fi/fi/ajankohtaista/suurteholatausasemien-100-kw-kasvu-jatkuu-voimakkaana

Traficom. 2021. Vaihtoehtoisten käyttövoimien ja polttoaineiden lataus- ja tankkauspisteitä. Viitattu 14.8.2022. Saatavissa: https://www.traficom.fi/fi/ajavaihtoehtoa/vaihtoehtoisten-kayttovoimien-ja-polttoaineiden-lataus-ja-tankkauspisteita

Tutorials Point. TypeScript – Overview. Viitattu: 22.8.2022. Saatavissa: https://www.tutorialspoint.com/typescript/typescript_overview

TutorialsTeacher.com. What is Node.js? Viitattu: 22.8.2022. Saatavissa: https://www.tutorialsteacher.com/nodejs/what-is-nodejs

TutorialsTeacher.com. Node.js Module. Viitattu: 22.8.2022. Saatavissa: https://www.tutorialsteacher.com/nodejs/nodejs-modules

Yle. 2022. EU:n esitys: Sähköauton latausaseman pitäisi vuosikymmenen loppuun mennessä löytyä 60 kilometrin etäisyydeltä isosta osasta Suomea. Viitattu 4.9.2022. Saatavissa: https://yle.fi/uutiset/3-12478130

Yle. 2022. Julkinen liikenne sähköistyy vauhdilla – Tampereen uusien sähköbussien toimintavarmuuden luvataan olevan pakkasellakin dieseleitä parempi. Viitattu: 14.8.2022. Saatavissa: https://yle.fi/uutiset/3-12472972