Tampereen yliopisto

Valtteri Yli-Karro

LUOTETTAVAN TIEDONSIIRRON TCP/IP-PROTOKOLLAT

Kandidaatintyö
Informaatiotekniikan ja viestinnän tiedekunta (ITC)
Tarkastaja: Lehtori Juha Vihervaara
Joulukuu 2022

TIIVISTELMÄ

Valtteri Yli-Karro: Luotettavan tiedonsiirron TCP/IP-protokollat
Kandidaatintyö
Tampereen yliopisto
Tieto- ja sähkötekniikan kandidaattiohjelma
Joulukuu 2022

Luotettava tiedonsiirto on yksi internetin tärkeimmistä ominaisuuksista. Se mahdollistaa digitaalisen tiedonsiirron siten, että lähettäjä ja vastaanottaja voivat olla varmoja siitä, että pyydetyt tiedot saapuivat perille kokonaisina ja virheettöminä. Nykyajan tiedonsiirtoon pitää usein vielä sisällyttää salaus, jotta mahdolliset hyökkääjät eivät pääse lähetettävään tietoon käsiksi.

Työssä tarkastellaan TCP/IP-pinon (Transmission Control Protocol / Internet Protocol) kolmea tiedonsiirtoprotokollaa, jotka toteuttavat luotettavan tiedonsiirron. Nämä protokollat ovat TCP (Transmission Control Protocol), SCTP (Stream Control Transmission Protocol) ja QUIC. Näiden keskeiset ominaisuudet esitetään yksityiskohtaisesti, ja niiden kautta tarkastellaan eri tapoja toteuttaa luotettavaa tiedonsiirtoa. Jokaisen protokollan kohdalla tutkitaan, miten yhteys päätelaitteiden välille muodostetaan ja miten tiedonsiirto tässä muodostetussa yhteydessä tapahtuu.

Protokollat on valittu niin, että niiden toteutustavat tai käyttötarkoitukset eroavat toisistaan. Pääpainona työssä on internetin tiedonsiirto, johon Googlen kehittämä QUIC on uusi varteenotettava tulokas. QUICin tutkimista tukevat TCP:n ja SCTP:n esittelyt, sillä QUIC jakaa saman käyttötarkoituksen TCP:n kanssa, mutta sen toimintaperiaate on lähempänä SCTP:tä.

Tutkittavia protokollia vertaillaan toisiinsa niistä esitettyjen ominaisuuksien avulla. Vertailun tueksi on haettu erilaisia tutkimuksia ja artikkeleja. Vertailussa käy ilmi, miksi QUICin kehitys on alun perin aloitettu, vaikka jo olemassa ollut SCTP toteuttaa pintapuolisesti vastaavanlaista tiedonsiirtoa. Tämä avaa näkökulmia tulevaisuuden tiedonsiirtoprotokollien suunnitteluun. Yhteyden muodostamisesta johtuvan viiveen vähentäminen on ollut merkittävässä osassa QUICin määrittelyä jo ensimmäisistä julkaistuista artikkeleista lähtien. Viivettä pyritään vähentämään sisällyttämällä mahdollisimman paljon ominaisuuksia kuten turvallisuutta jo heti ensimmäisiin paketteihin, jolloin nämä ominaisuudet eivät tarvitse omia erillisiä kättelyitään.

Avainsanat: TCP/IP, TCP, SCTP, QUIC, luotettava tiedonsiirto

ABSTRACT

Valtteri Yli-Karro: Reliable TCP/IP communication protocols
Bachelor's thesis
Tampere University
Bachelor's degree in Computing and Electrical Engineering
December 2022

Reliable data transfer is one of the most important qualities of the internet. It enables digital data transfer so that the sender and receiver can be certain that the requested data has been delivered in full and undamaged. In today's world, data transfer will usually need to include encryption so that possible attackers can't access the information being transmitted.

This thesis takes a look at three different data transfer protocols that provide reliable data transfer in the context of the TCP/IP-stack (Transmission Control Protocol / Internet Protocol). These protocols are TCP (Transmission Control Protocol), SCTP (Stream Control Transmission Protocol), and QUIC. The main features of these protocols are presented in detail and through them, observations are made on how reliable data transfer can be achieved. It is observed how the connection between endpoints is formed and how data transmission is implemented in this connection for each protocol separately.

The protocols are chosen so that their implementations or use cases differ from one another. The main focus of this thesis is data transfer on the Internet for which QUIC implemented by Google is a new noteworthy newcomer. The introductions to TCP and SCTP support investigating QUIC, as QUIC shares a similar use case with TCP but it delivers data more like SCTP.

The protocols are compared to each other by their represented qualities. To support this comparison, various studies and articles have been used. QUIC's design principles and rationale are found in the comparison and also why QUIC has been developed in the first place while SCTP seems to be implemented in the same way on the surface. This opens up views for the future design of data transfer protocols. Reducing the delay caused by initiating a connection has been a major reason for implementing the QUIC protocol ever since the first design rationale articles. Delay is being reduced by including many features such as security straight to the first sent packets so that these won't be needing their separate handshakes.

Keywords: TCP/IP, TCP, SCTP, QUIC, reliable data transfer

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

APNIC	Asia-Pacific Network Information Centre
ARP	Address Resolution Protocol
ARQ	Automatic Repeat reQuest
DNS	Domain Name System
GBN	Go-Back-N
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISN	Initial Sequence Number
MTU	Maximum Transmission Unit
NAT	Network Address Translation
RFC	Requests for Comments
SCTP	Stream Control Transmission Protocol
SSH	Secure Shell
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol / Internet Protocol
TLS	Transport Layer Security
TSN	Transmission Sequence Number
UDP	User Datagram Protocol

1. JOHDANTO

Tiedonsiirto on nykymaailmassa jatkuvasti tärkeämmässä ja vaikutusvaltaisemmassa roolissa. Liikutettavan tiedon määrä kasvaa jatkuvasti internetin käyttäjien ja laitteiden lisääntyessä [9]. Myös turvallisuusaspekti on tullut suurempien yhteys- ja laitemäärien myötä jatkuvasti tärkeämmäksi teemaksi internetin tiedonsiirrossa. Tämä lisää oman osansa tiedonsiirtoprotokollien tehokkuuteen ja toteutustapaan.

Tiedonsiirtoon on kehitetty useita protokollia, joista tässä työssä käydään läpi kolmea TCP/IP-mallin protokollaa: TCP:tä, SCTP:tä ja QUICia. Näistä TCP ja SCTP toimivat mallin kuljetuskerroksella ja QUIC sovelluskerroksella. Nämä on esitelty järjestyksessä vanhimmasta uusimpaan, ja kaikki niistä eroavat toisistaan ominaisuuksiltaan ja osittain myös käyttötarkoituksiltaan. Yhteistä näille kuitenkin on, että ne ovat luotettavia, eli ne varmistavat tiedonsiirron oikeellisuuden ja eheyden. Tiedonsiirron tietyt protokollat ovat vakiintuneet jo internetin varhaisessa vaiheessa, joten uuden rakentaminen pohjautuu yleensä jollain tavalla näihin vanhoihin protokolliin. Tämä tullaan huomaamaan luvussa 3.3.

Työn tavoitteena on selvittää, minkälaisia toteutuksia tiedonsiirrolle voi olla ja käydä läpi tarkemmin näitä kolmea edellämainittua protokollaa. Työssä pyritään myös tietyssä määrin vertailemaan näiden protokollien ominaisuuksia keskenään.

Luvussa 2 käydään läpi TCP/IP-mallia ja sen eri kerroksia. Samassa luvussa kerrotaan myös, mitä luotettava tiedonsiirto tarkoittaa tämän työn kontekstissa ja esitellään ratkaisumalleja, joilla sitä voidaan toteuttaa.

Luvussa 3 esitellään protokollia ja käydään niitä tarkemmin läpi. Jokaisen protokollan kohdalla kerrotaan, miten ne rakentuvat ja miten luotettava tiedonsiirto niissä käytännössä tapahtuu. Luku 3.1 on TCP:lle, 3.2 SCTP:lle ja 3.3 QUICille. Luvussa 4 vertaillaan luvussa 3 käytyjä prokollia toisiinsa ominaisuuksien, käyttökohteiden ja tutkittujen nopeuksien perusteella.

2. LUOTETTAVAN TIEDONSIIRRON PERUSPERIAATTEET

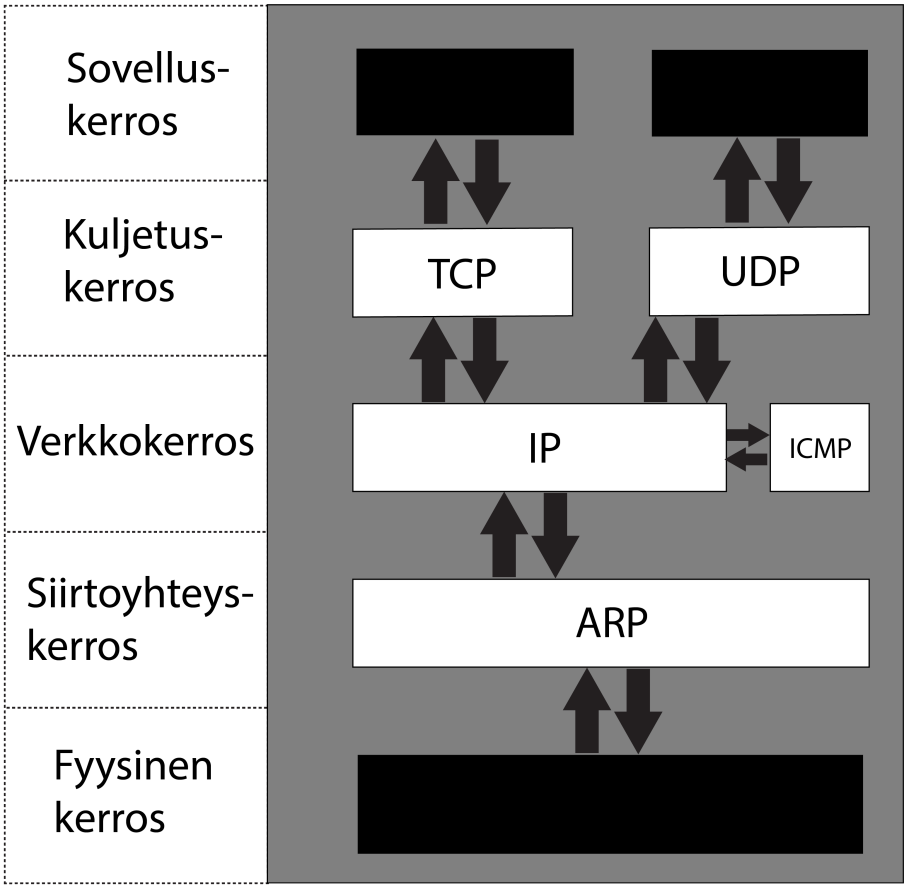
Luotettava tiedonsiirto tarkoittaa tietotekniikassa tapaa siirtää digitaalista tietoa niin, että voidaan olla varmoja tiedon välittymisestä. Käytännössä tämä tapahtuu käyttämällä erilaisia varmistusmekanismeja, joiden avulla tiedon lähettäjä ja vastaanottaja voivat varmistua siitä, että sanoma on saatu muuttumattomana perille.

2.1 TCP/IP

TCP/IP (Transmission Control Protocol / Internet Protocol) tarkoittaa useamman protokollan yhdistelmää, jolla toteutetaan tiedosiirtoa verkon eri laitteiden välillä. TCP/IP onkin vakiintunut internetin verkkosysteemiksi monen kilpailijan edelle. Yksi TCP/IP:n eduista on, että se ei ole millään tapaa laitteistoriippuvainen, eli se toimii lähtökohtaisesti kaikilla internetiin kytkettävillä laitteilla suoraan. TCP/IP on myös hyvin skaalautuva järjestelmä, ja se toimii hyvin erikokoisissa verkoissa. Nämä edellämainitut ominaisuudet mahdollistavat erilaiset TCP/IP:n päälle rakennetut protokollat, joita tässäkin työssä tarkastellaan.

Internetiin liittyvät standardit ja esitykset on yleisesti määritelty IETF:n (Internet Engineering Task Force) julkaisemissa RFC-asiakirjoissa (Requests for Comments). Näissä on määritelty laajasti erilaisia asioita kuten jo vakiintuneita standardeja tai esimerkiksi muutosehdotuksia johonkin muuhun RFC:hen. Jotkut RFC:t saattavat olla ihan vain ohjeita jonkin asian käyttöön, esimerkiksi RFC 1180, joka kuvailee TCP/IP-käyttöohjeet.

TCP/IP-protokollaperhe sisältää esimerkiksi seuraavat verkkoprotokollat: IP (Internet Protocol), ICMP (Internet Control Message Protocol), ARP (Address Resolution Protocol), UDP (User Datagram Protocol) ja TCP (Transimission Control Protocol). Näistä tämän työn kontekstissa kiinnostaa lähinnä UDP ja TCP, jotka hoitavat TCP/IP:n tiedonsiirtopuolta ja toimivat mallin kuljetuskerroksella. Muut TCP/IP-pinon protokollat toimivat taasen lähtökohtaisesti mallin alemmilla tasoilla, mutta luovat tärkeän pohjan TCP:n ja UDP:n toiminnalle. Tämä nähdään kuvasta 2.1, josta on jätetty pois sovelluskerroksen ja fyysisen kerroksen osat, sillä ne eivät suoraan ole TCP/IP-protokollapinossa, vaikka TCP/IP usein mielletäänkin 5-kerroksiseksi malliksi. [12]

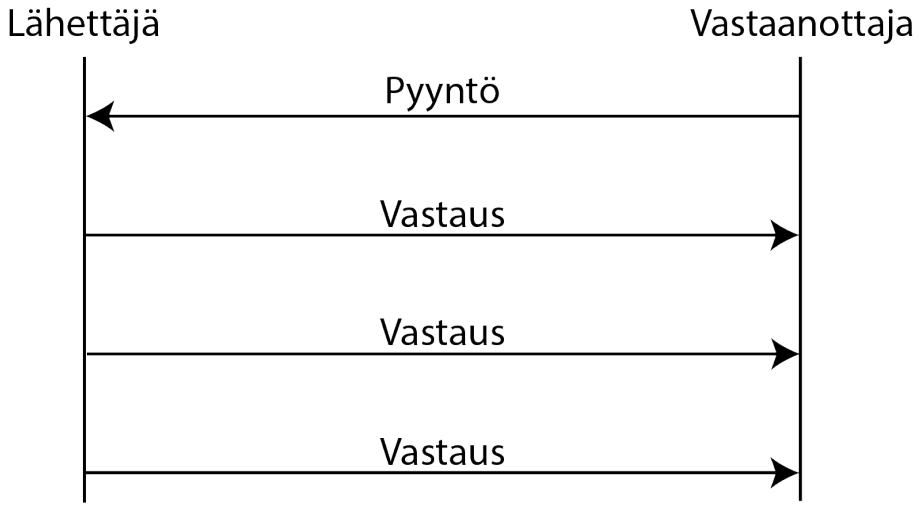


Kuva 2.1. *TCP/IP-protokollat kerroksittain kuvattuna. Muokattu lähteestä [12].*

Sovelluskerroksella toimivia sovelluksia ovat esimerkiksi TCP:n päällä toimiva SSH (Secure Shell) ja UDP:n päällä toimiva DNS (Domain Name System). Fyysiselle kerrokselle taasen kuuluvat esimerkiksi kuparijohdot, optiset kaapelit ja radioaallot, eli nimensä mukaisesti fyysiset datansiirtomediat. Nämäkään eivät kuitenkaan määritelmällisesti kuulu edellä mainittuihin TCP/IP-pinon protokolliin.

Suunnittelulähtökohtiensa vuoksi UDP on yleisesti mielletty epäluotettavaksi protokollaksi, kun taas TCP on rakennettu luotettavaksi. Jotta voidaan ymmärtää luotettavan tiedosiirron periaatteet, tarkastellaan tässä työssä myös UDP:ta. TCP-yhteyttä käydään tarkemmin läpi luvussa 3.1.

UDP toimii yhteydettömänä protokollana, joka toteuttaa vain minimaaliset virheentarkistukset. Se ei siis käytännössä rakenna kovinkaan paljoa lisää IP:n päälle. UDP ei muun muassa toteuta kättelyitä ennen tiedonsiirron aloittamista, vaan aloittaa tiedonsiirron suoraan saatuaan sovelluskerroksen sovellukselta lähetettävän tiedon [10]. Tämä voidaan nähdä kuvassa 2.2, jossa vastaanottaja lähettää pyynnön ja lähettäjä alkaa lähettämään paketteja, "vastauksia", välittömästi pyynnön saatuaan sen enempää varmistelematta.



Kuva 2.2. *UDP-protokollan tiedonlähetys vastaanottajan pyynnöstä. Muokattu lähteestä [28].*

Vaikka UDP ei olekaan tämän työn keskiössä, on sen toimintaperiaate tärkeä ymmärtää, sillä nykyään myös sen päälle on rakennettu luotettavan tiedonsiirron protokollia onnistuneesti. Yksi näistä esimerkeistä on QUIC, josta kerrotaan lisää luvussa 3.3.

2.2 Luotettava tiedonsiirto

Vaikka tässä työssä on tähän mennessä painotettu kuljetuskerroksen protokollia, on huomattava, että luotettavan tiedonsiirron protokollien suunnittelussa ja toteutuksessa tulee huomioida kaikki kerrokset TCP/IP-mallista. UDP-esimerkistä kuvassa 2.2 voitiin jo päätellä, että esimerkiksi IP ei verkkokerroksen protokollana ole itsessään luotettava. Sen päälle pitää rakentaa ylemmillä kerroksilla luotettavuutta lisääviä rakenteita, kuten tässä työssä luvussa 3 mainitut protokollat tekevät. Luotettavan protokollan pitää siis huomioida kaikki mahdolliset ongelmat, mitä sitä alemmat tasot voivat sille aiheuttaa. Esimerkkinä tällaisesta on tiedon korruptoituminen tai bittijärjestyksen muuttuminen. [10]

Eräs tapa järjestellä tiedonsiirron luotettavuus on niin sanottujen vahvistusviestien (acknowledgement, ACK) lähettäminen aina jokaisen viestin jälkeen. Näin lähettäjä saa aina vahvistuksen, että viesti on tullut perille. Mikäli tämä toteutetaan niin, että vastaanottaja toistaa aina viestin takaisin lähettäjälle, on mahdollista huomata, mikäli viestit muuntuvat matkan varrella, esimerkiksi fyysisen tason häiriöistä johtuen. Häiriöiden huomaamiseen voidaan käyttää tarkistussummia, jotka lähetetään viestin mukana. ARQ (Automatic Repeat reQuest) on virheenhallintametodi, joka toteuttaa kontrollin viestien tarkistuksille, virheen havainnoinnille ja tarvittaessa uudelleenlähetykselle.

Edellisessä kappaleessa kuvatut protokollat ovat kuitenkin usein hitaita, mikäli jokaisen paketin jälkeen tarvitsee odottaa kuittausta. Tämän vuoksi modernit protokollat luovatkin niin kutsutun kanavan (pipeline), jota pitkin voidaan samanaikaisesti lähettää edestakaisin

vahvitus- ja datapaketteja. Näille löytyy kahta erilaista toteutustapaa: GBN (Go-Back-N) ja valikoiva toisto (selective repeat) protokollat, jotka ovat ARQ:n alatyyppejä.

GBN toimii niin, että paketteja voidaan lähettää tietty määrä (N) ennen kuin vastaanottajalta vaaditaan vahvistusviesti. Tämä toteutetaan käyttämällä pakettien järjestysnumeroita, joista vastaanottaja pitää kirjaa. Mikäli vastaanottaja saa virheellisen paketin, lähettää se vahvistusviestin viimeisen oikean paketin järjestysnumerolla. Tällöin lähettäjä laittaa siitä lähtien seuraavat paketit uudelleenlähetykseen. Mikäli kaikki paketit ovat oikein, vastaanottaja lähettää vahvistuksen aina tietyin sovituin välein (N).

Edellä esitelty GBN kuitenkin saattaa aiheuttaa paljon ylimääräistä lähettämistä, etenkin jos tälle tavalle tärkeä vahvistusviesti hukkuu matkalle. Tämä käy toteen etenkin, jos yhteys on heikko jo alun alkaen. Valikoiva toisto pyrkii estämään tätä tapahtumasta asettamalla jokaiselle paketille järjestysnumeron. Se pyytää vain tiettyjä paketteja uudestaan, mikäli ne ovat vaurioituneet. Koko pakettivirtaa ei siis lähetetä uudestaan tietystä vaurioituneesta paketista lähtien, vaan näin tehdään vain tarvittaville paketeille. [26]

3. LUOTETTAVAN TIEDONSIIRRON TCP/IP-PROTOKOLLAT

Luotettavaa tiedonsiirtoa varten on kehitetty useita protokollia, joista tässä työssä käydään tarkemmin läpi kolmea. Eri protokollilla on erilaiset toimintaperiaatteet ja käyttötarkoitukset, mutta kaikkien niiden lähtökohtana on kuitenkin varmistaa tiedonsiirto luotettavasti. Tämä tarkoittaa, että tietoa ei kadoteta varmistaen, että vastaanottaja saa kaiken tiedon eheänä perille.

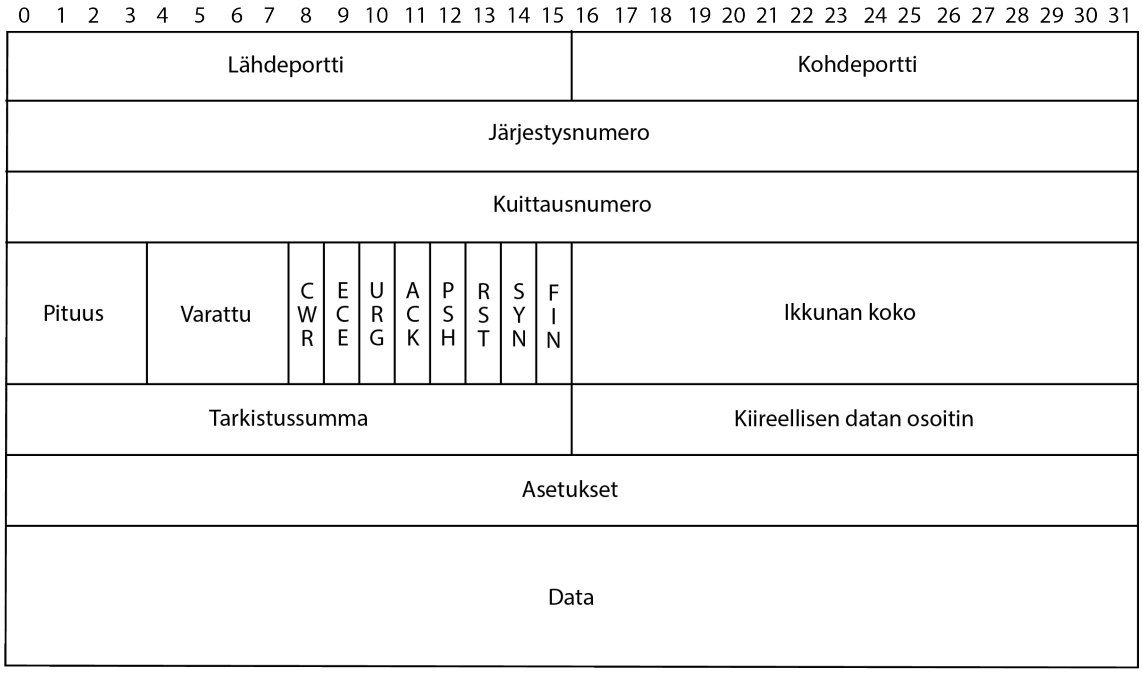
3.1 TCP

TCP on luotettavan tiedonsiirron perusprotokollia, ja se on myös TCP/IP-pinon oleellinen osa. Protokollaa voidaan myös käyttää omana itsenään, ja se onkin erittäin suosittu protokolla moneen käyttötarkoitukseen, joissa tarvitaan luotettavaa tiedonsiirtoa. Hyvänä esimerkkinä on, että valtaosa internetissä tapahtuvasta tiedonsiirrosta tapahtuu juuri TCP-protokollan avulla. TCP:lle ominaista ovat sen yhdeyden alustamiseen käytettävä kolmivaiheinen kättely ja pakettien välissä lähetetyt ACK-viestit, joilla varmistetaan viestien virheetön vastaanotto. TCP:n määrittely tapahtuu dokumentissa RFC 9293, johon on kerätty erinäisistä aiemmista dokumenteista oleellinen tieto. RFC 9293 on julkaistu vuonna 2022, vaikka TCP itsessään on määritelty jo vuonna 1981 dokumentissa RFC 793 [16].

TCP:n lähtökohtana on tarkistaa pakettien kadottamiset järjestysnumeroiden perusteella ja tarkistaa lähetykset virheiden varalta tarkistussummia hyväksikäyttäen. Näitä tarkistuksia toteutaan jokaiselle TCP-kehykselle (segment). TCP-kehys koostuu otsikosta (header) ja toimitettavasta datasta. TCP:n kehysrakenne on esitetty kuvassa 3.1.

Jo itse otsikko sisältää useita eri osia, joista suurin osa on pakollisia, mutta myös vapaavalintaisia asetuksia voi antaa. Otsikko on TCP:ssä oleellinen osa luotettavaa tiedonsiirtoa, ja sillä pidetäänkin kirjaa monenlaisista asioista. Eri kentät ovat havaittavissa kuvasta 3.1, jossa otsikko on kaikki muut kentät paitsi alin eli data. Data on kuvassa TCP-paketin hyötykuorma (payload). Kuvan yläreunassa kulkee bittien numeroinnit. Jokainen kerros kuvassa merkitsee 32 bitin sarjaa. Näitä sarjoja kutsutaan yleisesti sanoiksi (words). Dataosuuden skaalaus kuvassa ei päde, vaan se voi olla huomattavasti suurempi kuin edellä

mainittu 32 bittiä.



Kuva 3.1. *TCP-protokollan kehys.*

Lähdeportti (source port) kertoo nimensä muikasesti, mistä portista paketti on tulossa. Kohdeportti (destination port) taasen on vastaanottavan tahon porttinumero. Nämä eivät sisällä tietoa IP-osoitteista, sillä tämä on IP:n tehtävä. Portit tulee kuitenkin olla tiedossa, jotta tiedetään, mille sovellukselle paketti on kohteessa tarkoitettu. Esimerkiksi HTTP-verkkopalvelimen (Hypertext Transfer Protocol) oletusarvoinen portti on suojaamattomille yhteyksille 80 ja suojatuille yhteyksille 443. Tämä toimii kyselyn kohdeporttina. Tietokoneet usein valitsevat lähdeportin sattumanvaraisesti eri verkko- yhteyksille ja odottavat palvelimen vastausta porttinumeroon, jota käytettiin kyselyssä lähdeporttina [4].

Järjestysnumero (sequence number) kertoo ensimmäisen data-oktetin järjestysnumeron. Data-osio on jaoteltu kahdeksan bitin sarjoihin, oktetteihin, joihin tässä viitataan. Oktetissa on siis käytännössä saman verran dataa kuin yhdessä tavussa. Mikäli SYN-lippu on asetettuna otsikossa, järjestysnumero ei osoita suoraan mihinkään datan palaan, vaan edelliseen numeroon ennen datan alkua. Tämän numeron lyhenteenä on ISN (Initial Sequence Number). Kuittausnumero on vastaavanlainen kirjanpitotyökalu kuin järjestysnumero. Se kertoo sen dataoktetin järjestysnumeron, jonka kuittauksen lähettäjä odottaa vastaanottavansa seuraavaksi.

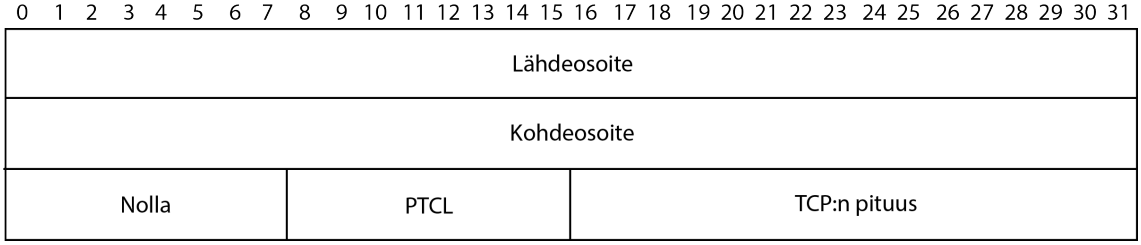
Pituus (Data Offset) on luku, joka ilmaisee otsikon pituuden ennen datan alkua sanoina. Tämä on tärkeää, sillä koko otsikon pituutta ei tarvitse täyttää, vaan esimerkiksi asetukset-kohta voidaan jättää käyttämättä. Käytännössä pituus siis kertoo vastaanottajalle, mistä kohtaa TCP-pakettia dataosuus alkaa.

Pituuden jälkeinen varattu-kenttä sisältää nimensä mukaisesti varattuja bittejä mahdollisia tulevaisuuden käyttötarkoituksia varten. Nämä pitää määritelmän mukaisesti asettaa arvoon 0, kunnes kehitetään ominaisuuksia, jotka vaativat niiden käyttöä.

Kontrolliliput tulevat varatun lohkon jälkeen. Nämä ovat yksittäisiä bittejä, jotka merkitsevät yksittäistä asiaa tai toimintamallia TCP-paketin käsittelyssä. Nämä ovat lähtökohtaisesti asetettu arvoon 0, mutta mikäli nämä pitää ottaa paketissa huomioon, vaihtaa paketin luoja arvoksi 1. [19]

Ikkunan koko on dataoktettien määrä lähetysikkunassa. Sen tehtävänä on rajoittaa datamäärän koko sopivaksi vastaanottajalle. Palvelin voi ilmaista ikkunan koko-osioissa, kuinka paljon sen puskuriin voidaan tallettaa dataa. Puskuri tässä tapauksessa rajoittaa kerralla lähetettävien dataoktettien maksimimäärää. On huomattava, että ikkunan koko kannattaa pitää mahdollisimman suurena, sillä otsikko vie TCP:ssä myös paljon bittejä dataosuuden ohella. Vastaanottajan puolella valinta keskittyy siihen, kuinka paljon se voi ottaa vastaan dataa kerrallaan ilman kuittausta. Myös verkon ruuhkaisuus voi vaikuttaa ikkunan koon määrittämiseen. [22]

Tarkistussumma (checksum) tarkistaa, että TCP-paketti on laadittu oikein ja se on saapunut perille vahingoittumattomana. Tarkistussumman laskemiseen otetaan mukaan paketin otsikko, dataosuus ja IP-protokollasta pseudo-IP-otsikko. Tähän otsikkoon kuuluvat lähdeosoite, kohdeosoite, 0-bitit, protokollan tyyppinumero (PTCL) ja TCP-paketin pituus, joka lasketaan erikseen. Pseudo IP-otsikko on esitetty kuvassa 3.2. Otsikon pituus on 96 bittiä.



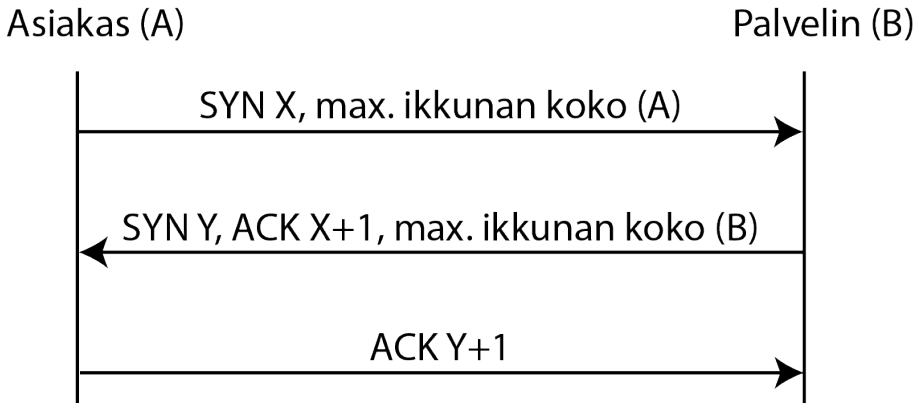
Kuva 3.2. *Pseudo-IP-otsikko. Muokattu lähteestä [19].*

Kiireellisen datan osoitin ilmoittaa, missä kiireellisen datan osoite tällä hetkellä menee suhteessa otsikon kehyksen järjestysnumeroon. Tätä kenttää tulee tulkita vain, jos URG-lippu on asetettu. Kiireellinen data toimitetaan vastaanottajalle ensimmäisenä, eli se siis poimitaan erikseen ja vasta sen jälkeen toimitetaan muu data.

Asetukset-kenttään voidaan merkitä otsikon loppuun mahdollisia ylimääräisiä asetuksia eli optioita, joita paketin hallinnassa tulee noudattaa. Asetukset ovat täysin vapaaehtoisia, joten asetuskentän pituus tulee huomioida myös pituus-kenttää täytettäessä. IANA (Internet Assigned Numbers Authority) ylläpitää listaa kaikista tällä hetkellä määritellyistä asetuksista [24].

TCP-yhteyden alussa tehdään kolmivaiheinen kättely, jonka aikana asiakas ja palvelin alustavat yhteyden datan lähettämistä varten. Kättely alkaa asiakkaan pyynnöstä palvelimelle. Pyynnössä lähetetään TCP-otsikko, jossa SYN-lippu on asetettu arvoon yksi. Asiakas lähettää myös järjestysnumeron X ja maksimiarvon kehyskoosta, jonka se on valmis vastaaottamaan.

Palvelin vastaa asiakkaan pyyntöön lähettämällä oman järjestysnumeronsa Y, kuittausnumeron asiakkaan järjestysnumeroon (X+1) sekä oman kehyskoon maksimiarvonsa. Kehyskooksi valitaan pienempi asiakkaan ja palvelimen toimittamista arvoista. Asiakas kuittaa tämän jälkeen saaneensa palvelimen viestin perille lähettämällä kuittausnumeron Y+1. Kättely on esitetty kuvassa 3.3.



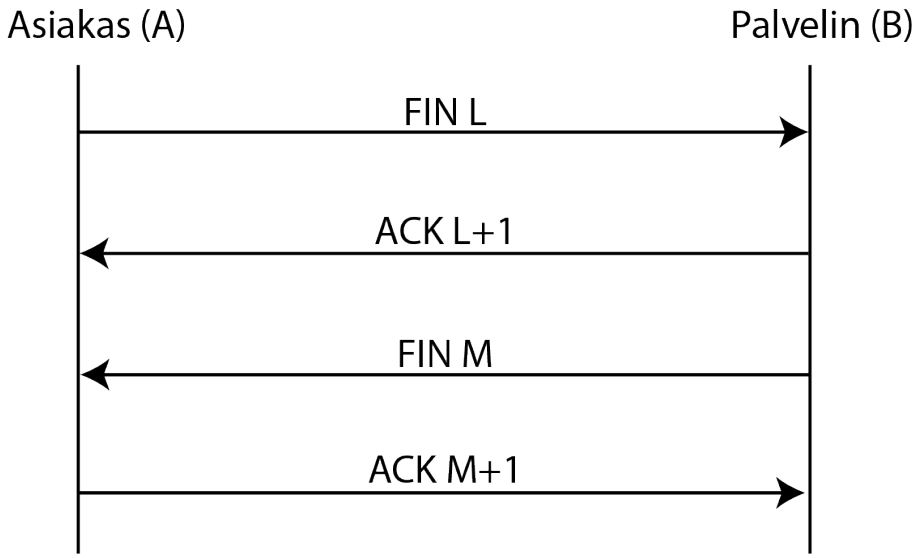
Kuva 3.3. *TCP-yhteyden alustaminen. Muokattu lähteestä [30].*

Mikäli yhteydestä halutaan turvallinen, tulee TCP-kättelyn jälkeen usein vielä erillinen TLS-kättely (Transport Layer Security). TLS-kättelyssä tehdään avaintenvaihto (key exchange), jotta paketit voidaan salata mahdollista verkkokuuntelua vastaan. Tämä kättely lisää yhteyden alustukseen neljä lisäaskelta [27].

Yhteyden avaamisen jälkeen tapahtuu itse datan siirto. Tämä toimii niin, että järjestysnumeroa, kuittausnumeroa ja tarkistussummaa tarkkailemalla pidetään huolta, että paketit saapuvat oikein perille. Jokaisen vastaanotetun paketin jälkeen vastaanottaja lähettää ACK-viestin lähettäjälle. Mikäli lähettäjä ei saa kuittausta jostain lähettämästään paketista, se lähettää paketin uudelleen vastaanottajalle.

Jokaisen paketin hyötykuorman (data) koko määräytyy sovitun kehyskoon mukaan. TCP-otsikon koko on 20 tavua, ja koko kehyksen maksimikoko on 65535 tavua (64 kilotavua). Pitää kuitenkin huomioida, että tämän kokoiset paketit eivät reaalimaailmassa toteudu juuri koskaan, sillä yleensä matalamman tason protokollat rajoittavat dataosuuden maksimikokoa (MTU, Maximum Transmission Unit). Esimerkiksi ethernet-standardin tapauksessa se on 1500 tavua. Tällöin TCP:lle jää ethernetin yli kulkevassa verkossa maksimidataosuudeksi 1480 tavua. [6]

Kun kaikki data on saatu siirrettyä, tehdään vielä nelivaiheinen yhteyden sulkeminen. Siinä asiakas lähettää paketin otsikossa järjestysnumeron L ja asettaa FIN-lipun arvoon 1. Palvelin vastaa tähän ACK-viestillä ja kuittausnumerolla L+1, jolla vahvistaa FIN-viestin tulleen perille. Palvelin lähettää vielä viimeisen FIN-viestin omasta puolestaan järjestysnumerolla M, jonka jälkeen asiakas vielä kuittaa sen vastaanotetuksi kuittausnumerolla M+1. TCP-yhteyden sulkeminen on esitetty kuvassa 3.4.



Kuva 3.4. *TCP-yhteyden sulkeminen. Muokattu lähteestä [30].*

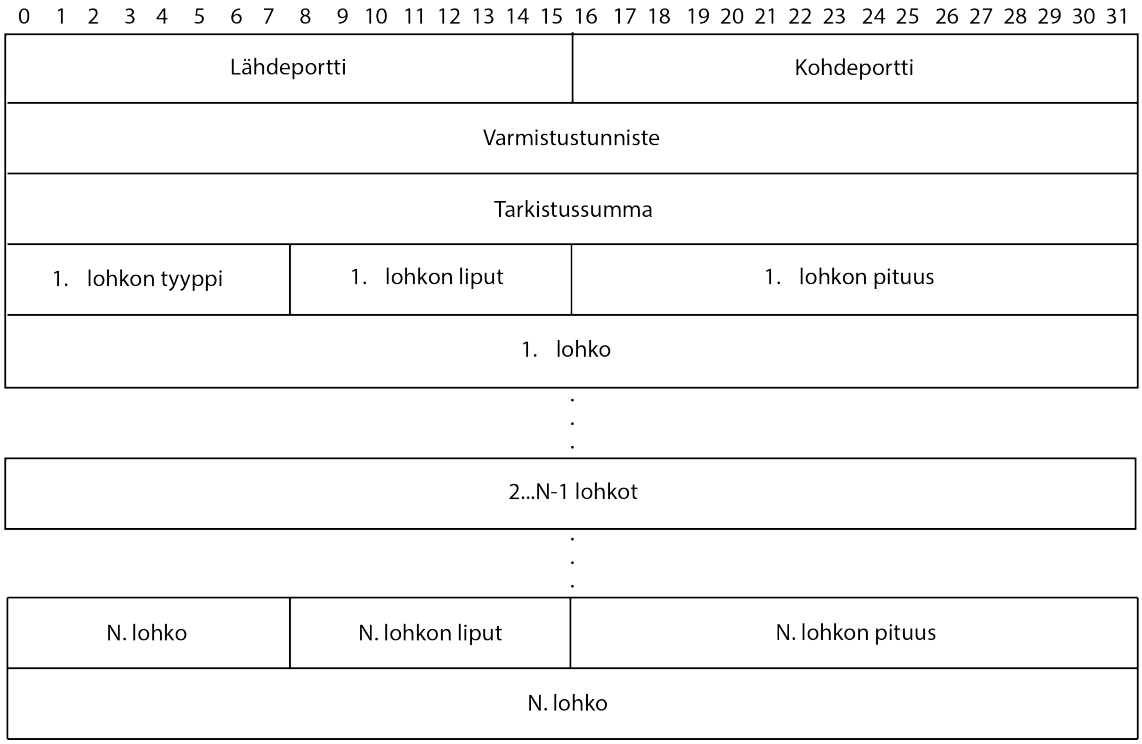
Yhteys saattaa myös katketa ennen kaiken datan lähettämistä esimerkiksi verkkokatkon vuoksi. Tilanne voi olla esimerkiksi sellainen, että paketti vaikuttaa siltä, ettei kuulu sillä hetkellä luodulle yhteydelle. Tällaisissa ongelmatapauksissa yleensä vastataan suoraan paketilla, jossa RST-lippu on ylhäällä, joka aiheuttaa yhteyden resetoinnin.

3.2 SCTP

SCTP (Stream Control Transmission Protocol) on protokolla, joka pyrkii ottamaan hyviä puolia sekä TCP:stä että UDP:stä. Se on kuitenkin täysin oma protokollansa, joka toimii samalla tasolla UDP:n ja TCP:n kanssa TCP/IP-mallissa eli kuljetuskerroksella IP:n päällä. SCTP:n suunnitteluperustana on puhelinverkon signalointijärjestelmän tiedonsiirto. SCTP:n lähtökohtana on ollut kitkeä ja parantaa TCP:ssä ilmenneitä heikkouksia ja haavoittuvuuksia.

SCTP on yhteydellinen protokolla, mutta sen yhteydestä puhutaan yleisesti termillä SCTP-assosiaatio (SCTP association). Assosiaatio tapahtuu kahden päätepisteen välillä. Se pitää sisällään protokollan tilan ja mitä reittiä pitkin päätepisteet voivat kommunikoida toistensa kanssa. Tämä reitti voi koostua useammasta IP-osoitteesta, joiden läpi yhteys kulkee. Kahdella päätepisteellä voi olla vain yksi SCTP-assosiaatio yhtäaikaisesti.

SCTP-yhteys muodostetaan lähettämällä SCTP-otsikko (header). Tämän jälkeen lähetetään kaikki tarvittavat lohkot (chunk), joita voi olla niin paljon kuin tarvitaan. Kehyksessä on määritelty kentät lähdeportti, kohdeportti, varmistustunniste ja tarkistussumma. Kaikille lohkoille yhteisiä kenttiä ovat tyyppi, liput ja pituus. Näiden jälkeen tulevat toiminnalliset lohkot, jotka voivat sisältää vielä ylimääräisiä kenttiä. Geneerinen SCTP-paketti on esitetty kuvassa 3.5.



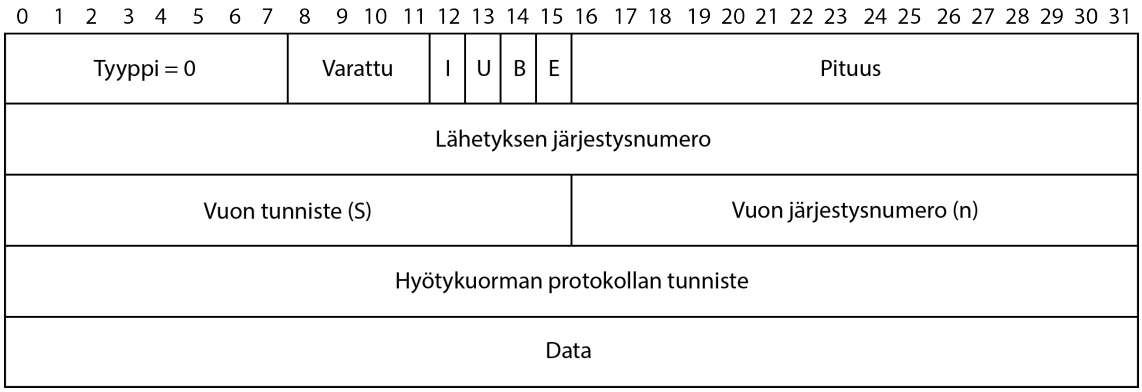
Kuva 3.5. *SCTP-protokollan geneerinen kehys.*

Lähdeportti ja kohdeportti ovat vastaavat kentät kuin TCP:ssä. Ne määrittävät lähettäjän ja vastaanottajan porttinumerot, joita käytetään määrittämään, mihin yhteyteen paketti liittyy. RFC 9260 kertoo erikseen, että 0-porttinumeroita ei saa käyttää kummassakaan kentässä. Myös tarkistussumma toimii vastaavanlaisessa roolissa kuin TCP:ssä. Sen tehtävä on varmistaa, että paketti on saapunut perille ehjänä.

Varmistustunnisteen (verification tag) avulla varmistetaan, että paketin lähettäjä on oikea. Alkuun tämä asetetaan samaan arvoon kuin yhteyden alustavan tahon toimittama arvo. Tähän on kuitenkin muutama poikkeus, jotka liittyvät siihen, minkä tyyppinen lohko toimitetaan paketin mukana. INIT-lohkossa varmistustunniste asetetaan arvoon 0. SHUTDOWN COMPLETE -lohkossa varmistustunniste kopioidaan sitä ennen vastaanotetusta SHUTDOWN ACK -paketista. Mikäli paketissa on ABORT-lohko, voidaan varmistustunniste kopioida paketista, joka aiheutti ABORT-paketin lähetyksen.

Yksi oleellisimmista lohkotyypeistä SCTP:lle on DATA-lohko, jonka tehtävänä on sen nimensä mukaisesti kuljettaa tietoa. Datalohkon tyyppinumero on määritelmän mukaan 0.

Kuten muillakin lohkoilla, DATA-lohkon lohkotyyppi on määritelty selvästi, ja sen pitää sisältää tietyt tiedot. DATA-lohkossa nämä tiedot ovat tyyppi, kontrolliliput, pituus, lähetyksen järjestysnumero, vuon tunniste, vuon järjestysnumero, hyötykuorman protokollan tunniste ja itse lähetettävä data. DATA-paketin lohko-osio on esitetty kuvassa 3.6.



Kuva 3.6. *SCTP-protokollan datalohko. Muokattu lähteestä [18].*

Lohkon tyyppinumeron jälkeen tulevat varatut bitit, jotka tulee asettaa arvoon 0, ja niitä ei tule vastaanottavassa päässä huomioida. Nämä ovat, kuten TCP:ssä, varattuja bittejä tulevaisuuden kontrollibittejä eli lippuja varten. Liput tulevat varattujen bittien jälkeen. Kirjoitushetkellä niitä on käytössä 4: I (Immediate), U (Unordered), B (Beginning) ja E (Ending). I ja U -kontrollibitit määrittävät eri toimintatapoja pakettien käsittelylle, kun taas B ja E kertovat datan alun ja lopun.

Pituus ilmaisee lohkon pituuden tavuina. Pituus lasketaan tavuina tyyppinumerosta eli alusta alkaen. Ilman mitään hyötykuormaa (data-kenttä) pituus saa siis arvon 16. Tällainen arvo ei kuitenkaan ole hyväksytty, sillä DATA-lohkon datakentässä pitää olla vähintään tavun verran bittejä. Minimiarvo on siis oikeasti 17, jolloin hyötykuormaa olisi 1 tavu.

Lähetyksen järjestysnumero eli TSN (Transmission Sequence Number) kertoo nimensä mukaisesti lähetettävän lohkon järjestysnumeron. Kun maksimiarvo 4294967295 (32 bittiä) saavutetaan, jatketaan laskemista nollasta ylöspäin.

Vuon tunniste (Stream Identifier) kertoo, mihin vuohon (stream) lohko kuuluu. Se siis käytännössä varmistaa, että paketti menee sinne mihin kuuluu. Vuon järjestysnumero (Stream Sequence Number) on vastaavantyyppinen arvo kuin vuon tunniste. Tätä käytetään seuraamaan tietyn vuon sisällä lähetetyn datan järjestystä. Tätä tarvitaan esimerkiksi, kun dataa on pilkottu osiin lähetystä varten.

Hyötykuorman protokollan tunniste (Payload Protocol Identifier) välittää ylemmän tason protokollille tai sovelluksille tiedon, mihin datalohko siellä kuuluu. SCTP itsessään ei tätä lohkoa käytä tai muuta, mutta se on silti pakollinen kaikille DATA-tyyppisille lohkoille, koska se varmistaa, että ylemmän tason sovellukset osaavat hyödyntää datan oikein. Mikäli ylempi kerros ei ole määritellyt mitään tunnisteta, lähetetään nämä bitit arvolla 0.

Itse data-kenttä voi olla vaihtelevan mittainen. Kuitenkin siihen pitää lisätä nollia perään niin, että datan määrä on neljällä tavulla jaollinen. Näitä ylimääräisiä nollia ei lasketa pituus-kohtaan mukaan. [18]

Toinen SCTP:lle tärkeä paketti on yhteyden avaamisessa käytettävä INIT-lohko. SCTP-assosiaatio avataan asiakkaan pyynnöstä. INIT-paketti on ensimmäinen lähetettävä paketti, ja sen tyyppinumero on 1. INIT-lohko on esitetty kuvassa 3.7.

Kuva 3.7. *SCTP-protokollan INIT-lohko. Muokattu lähteestä [18].*

INIT-lohkon kolme ensimmäistä kenttää ovat vastaavat kuin DATA-lohkossa. Lippuja ei ole INIT-pakettiin määritelty lainkaan, vaan ne ovat varattuja bittejä mahdollisia tulevaisuuden käyttötarkoitusta varten. Varatut lippubitit tulee asettaa lohkossa arvoon 0.

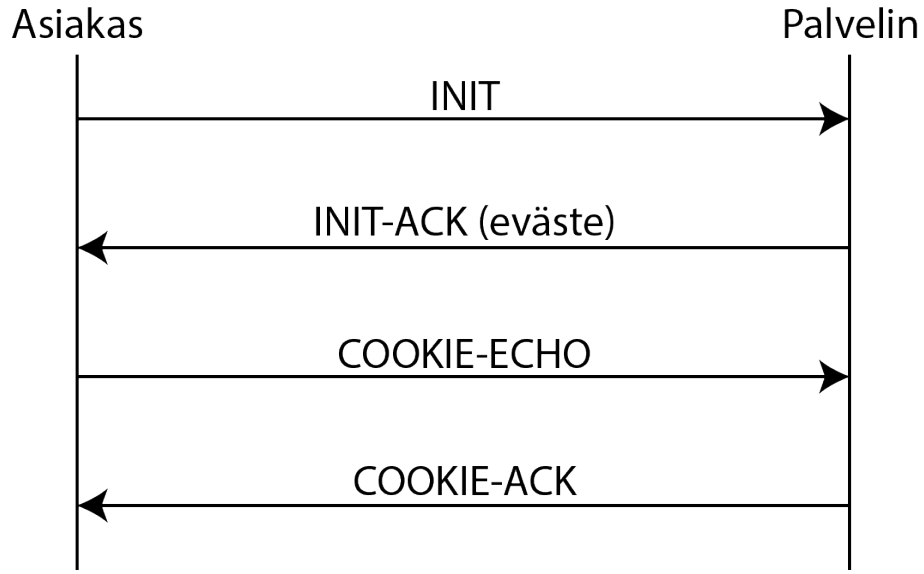
Alustusmerkki (Initiate tag) voi olla mikä tahansa luku paitsi 0, ja se kannattaa yleisesti valita sattumanvaraisesti, jotta mahdollisilta hyökkäyksiltä vältytään. Vastaanottajan tulee liittää tämä tuleviin paketteihin mukaan, jotta tiedetään, mihin assosiaatioon paketti kuuluu. Alustusmerkissä määriteltyä arvoa ei tule luodun assosiaation aikana vaihtaa.

Mainostettu vastaanottoikkunan arvo (Advertised Receiver Window Credit) kuvaa, kuinka suuren puskurin (buffer) INIT-lohkon lähettäjä on varannut SCTP-assosiaatiolle. Arvo ei saa olla pienempi kuin 1500 tavua. Mikäli näin on, tulee paketti jättää huomiotta vastaanottavassa päässä. Assosiaation aikana tätä arvoa ei tulisi pienentää, mutta tarpeen tullen sitä voidaan muuttaa tai kasvattaa.

Ulos suuntaavien voiden lukumäärä (Number of Outbound Streams) ilmaisee kuinka monta vuota lohkon lähettäjä haluaa SCTP-assosiaatioon luoda ulospäin. Sisään suuntaavien voiden lukumäärä (Number of Inbound Streams) kertoo taasen kuinka monta vuota lohkon lähettäjä sallii vastaanottajan luovan itseään kohti. Kumpikaan arvo ei saa tässäkään tapauksessa olla 0. Vastaanottaja voi esittää eriävää määrää voita, jolloin valitaan esitetyistä arvoista pienempi.

Ensimmäinen järjestysnumero (Initial TSN) kertoo, mikä arvo otetaan ensimmäiseksi järjestysnumeroksi, jota ruvetaan seuraavissa lohkoissa kasvattamaan. Tämä alustaa esimerkiksi kuvassa 3.6 esiintyvän DATA-lohkon TSN:n. Alkuarvo on satunnaisesti valittu. Ensimmäisen järjestysnumeron jälkeiseen kenttään tulevat vapaavalintaiset parametrit. Tämä kenttä voidaan jättää täyttämättä, ja sen pituus tulee huomioida lohkon pituus -kentässä. [18]

SCTP-assosiaation avaus tapahtuu neliosaisella kättelyllä. Sen osia ovat INIT, INIT-ACK, COOKIE-ECHO ja COOKIE-ACK. Nämä kaikki ovat, kuten edelläkin on mainittu, omanlaisiaan lohkoja. SCTP-assosiaation nelivaiheinen kättely on esitetty kuvassa 3.8. Näitä erilaisia lohkoja ei käydä syvällisemmin läpi tässä työssä, mutta niiden määrittelyt löytyvät RFC 9260 -dokumentista.



Kuva 3.8. *SCTP-assosiaation nelivaiheinen kättely. Muokattu lähteestä [29].*

Asiakas lähettää ensin palvelimelle INIT-viestin, jonka tehtävä on ilmoittaa palvelimelle, että assosiaatio halutaan avata. Kun palvelin vastaanottaa INIT-viestin, lähettää se asiakkaalle takaisin INIT-ACK-paketin, joka sisältää assosiaatiota varten evästeen (cookie). Tätä evästettä hyödynnetään assosiaation ajan yhteyden tunnistamiseen. Se sisältää palvelimen tunnistetiedot, jotka on allekirjoitettu palvelimen salaisella kryptografisella avaimella (secret key).

Asiakkaan lähettämä COOKIE-ECHO yksinkertaisesti lähettää juuri vastaanotetun evästeen takaisin palvelimelle. Näin varmistetaan, että INIT-ACK-paketti on vastaanotettu oikein. Palvelin vahvistaa tämän käyttämällä salaista avaintaan. Tämän jälkeen palvelin vielä lähettää COOKIE-ACK-paketin asiakkalle, jonka jälkeen assosiaatio on avattu, ja datapakettien lähetys voi alkaa.

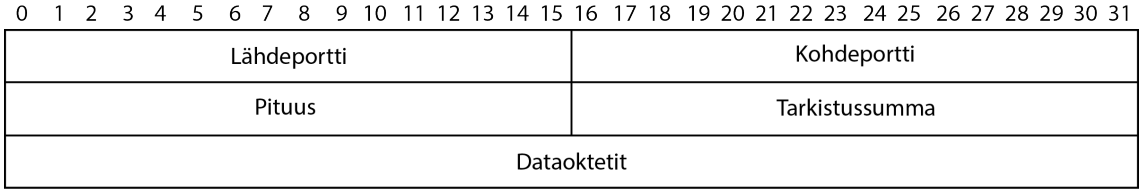
Assosiaation hallittu sulkeminen tapahtuu asiakkaan aloitteesta. Asiakas lähettää pal-

velimelle SHUTDOWN-paketin, jonka jälkeen palvelin vastaa tähän SHUTDOWN-ACK-paketilla, joka vahvistaa, että sulkemispaketti on vastaanotettu. Tämän jälkeen asiakas vielä lähettää yhden paketin, SHUTDOWN-ACK, minkä jälkeen yhteys on hallitusti suljettu. [20]

3.3 QUIC

QUIC on tässä työssä esiteltävistä protokollista selkeästi uusin. Protokollan kehitys alkoi vuonna 2012 Googlen toimesta. Sen tavoitteena on parantaa internetin loppukäyttäjäkokemusta vähentämällä sivujen lataamisen viivettä ja latausaikoja. Vuonna 2021 IETF julkaisi RFC 9000 QUICista. Poiketen muista tässä työssä esiteltävistä protokollista, QUIC ei ole lyhenne mistään, vaan toimii protokollan varsinaisena nimenä. [13]

QUIC pohjautuu UDP:hen, jota yleisesti pidetään epäluotettavana protokollana. UDP:ta esiteltiin aiemmin tässä työssä luvussa 2.1. Moni myöhemmin tässä luvussa esitelty QUICin ominaisuus perustuu tähän UDP:n päällä toimimiseen. UDP siis mahdollistaa QUIC-pakettien lähettämisen UDP-tietosähkeinä (datagram). UDP-tietosähkeeseen kuuluu otsikko, jonka rakenne on esitetty kuvassa 3.9. Otsikko lisätään jokaisen lähetettävän tietosähkeen alkuun.

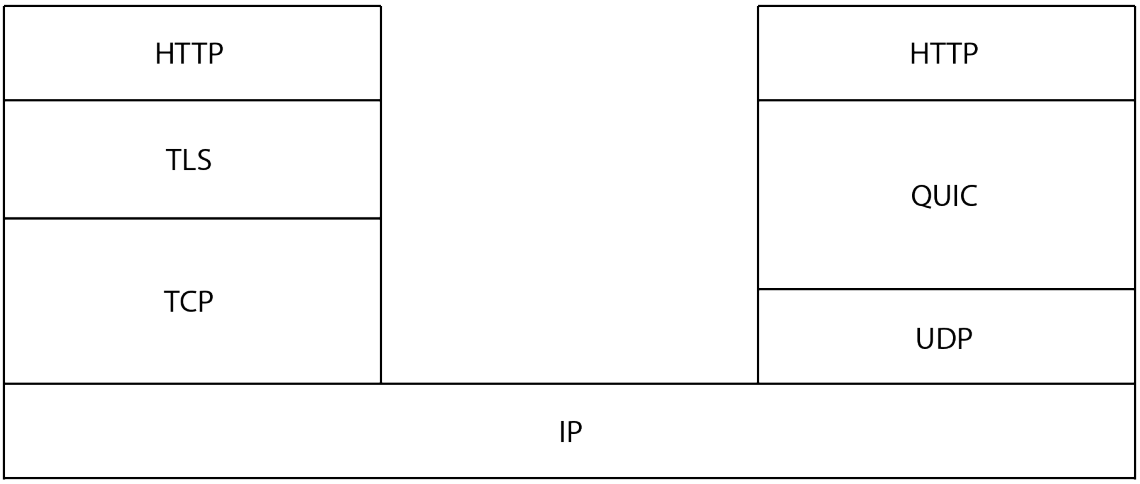


Kuva 3.9. *UDP-tietosähkeen otsikko.*

UDP-tietosähkeen otsikossa lähdeportti ja kohdeportti kertovat lähettäjän ja vastaanottajan porttinumerot, aivan kuten jo aiemmin tässä työssä esitellyissä TCP:ssä ja SCTP:ssä. Pieni ero kuitenkin tässä on: lähdeportti on UDP:ssa vapaaehtoinen, ja se tarvitsee ilmoittaa vain, mikäli sillä on merkitystä. Mikäli tätä ei tarvita, käytetään sen tilalla arvoa 0. Pituus ilmoittaa tietosähkeen koko pituuden eli dataoktettien määrän ja otsikon pituuden yhteenlaskettuna. Tarkistussumma lasketaan kuten TCP:ssä ja SCTP:ssä, ja sen tarkoitus on varmistaa, että paketti on reititetty oikeaan paikkaan. [15]

QUIC pyrkii vähentämään yhteyden muodostamiseen tarvittavaa aikaa TCP:hen pohjautuviin protokolliin verrattuna. Yksi QUICin tärkeistä sovelluskohteista on HTTP/3 (Hypertext Transfer Protocol 3), joka mahdollistaa internetissä nopeammat tiedonsiirrot pienemmällä viiveellä verrattuna HTTP:n vanhempiin toteutuksiin. HTTP/3 tarjoaa myös parempaa virheensietoa ja muita ominaisuuksia verrattuna vanhempiin HTTP/2 ja HTTP/1.1 -versioihin, jotka toimivat TCP:n avulla. Koska QUICin sunnittelulähtökohtana on käytännössä ollut vain HTTP/3, nojaa tämä kappale asioiden esittämiseen sen kautta. [7]

QUIC sisältää TLS:n version 1.3, joten sitä varten ei tarvita erillistä kättelyä kuten TCP:n kanssa, vaan se kulkee muiden QUICin UDP-tietosähkeiden mukana. Tämän ominaisuuden ansiosta kaikki QUIC-paketit ovat lähtökohtaisesti salattuja TLS:n avulla. Pinon pohjimmaisena tämän työn tarkasteluskaalassa toimii IP-protokolla verkkokerroksella, UDP tämän päällä kuljetuskerroksella ja QUIC sekä sen päällä toimiva HTTP sovelluskerroksella. QUIC- ja TCP-protokollapinot on esitetty kuvassa 3.10.



Kuva 3.10. *TCP-ja QUIC-protokollapinot. Muokattu lähteestä [8].*

UDP itsessään on hyvin yksinkertainen protokolla, joka huomattiin jo luvussa 2.1. QUICin pitääkin toteuttaa TCP:n ja TLS:n ominaisuudet sisällään sovelluskerroksella, jotta ominaisuudet voidaan täsmätä internetin tiedonsiirron vaatimiin ominaisuuksiin muun muassa ruuhkan-, häiriön- ja yhteydenhallinnan kannalta.

Yksi UDP:n käytön myötä tullut ominaisuus QUICiin on yhteystunniste (connection ID). Tämä on käytännössä pitänyt toteuttaa, sillä verkkolaitteet eri paikoissa kuljetusta saattavat tehdä osoitteenmuutoksia (NAT, Network Address Translation) matkalla, ja UDP:ta ei ole määritelty hallitsemaan tämän tuottamia muutoksia. Yhteystunnisteen avulla QUIC pitää kirjaa siitä, mihin yhteyteen tietyt paketit kuuluvat. Tämä myös mahdollistaa QUIC-pakettien keskeytymättömän lähetyksen, vaikka päätepisteen IP-osoite muuttuisikin kesken lähetyksen.

Vuot (stream) ovat QUICin paketinlähetykseen käyttämiä kanavia. Nämä ovat melko vastaavanlaisia kuin TCP-yhteydessä, mutta erona on, että näitä voi olla monta yhtäaikaisesti kahden päätepisteen välillä kuten SCTP:ssä. Tämä tarkoittaa, että eri datapaketit liikkuvat toisistaan riippumatta, ja yhden paketin katoaminen ei aiheuta koko lähetyksen keskeytymistä uudelleenlähetyksen ajaksi. Useamman vuon hyödyntäminen tarkoittaa myös sitä, että kun salaus on kerran alustettu yhteyttä varten, ei sitä tarvitse eri voille tehdä uudestaan joka kerta. [8]

QUIC-yhteys alustetaan asiakkaan toimesta alustavalla paketilla (initial packet). Tämä

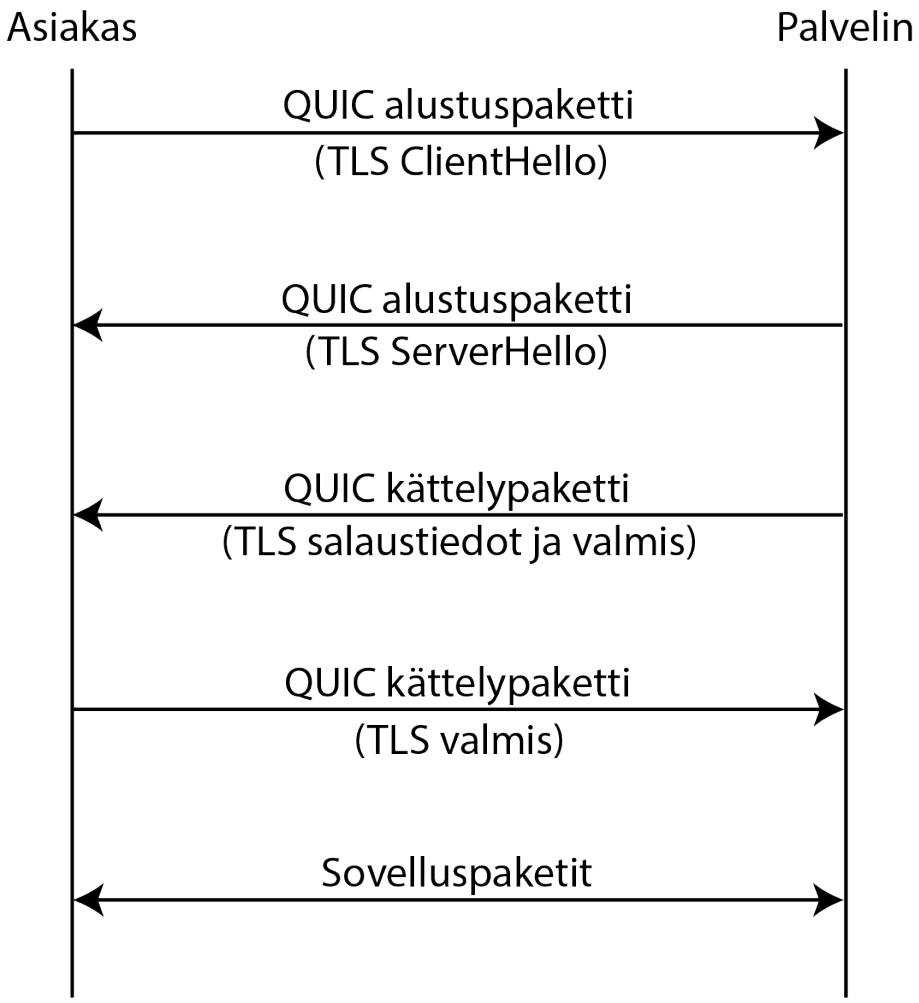
samainen paketti pitää sisällään TLS:n ClientHello-viestin, jolla aloitetaan salatun TLS-yhteyden luonti samalla kun QUIC-yhteys avataan.

Alustuspaketin mittaan lisätään täytteeksi (padding) nollia niin, että sen koko on vähintään 1200 tavua. Tälle tavujen lisäämiselle on kaksi merkittävää syytä. Ensinnäkin sillä varmistetaan, että reitillä jota pitkin yhteyden paketit tulevat kulkemaan, on riittävästi kapasiteettia hoitaa yhteys. Toisin sanoen reitin MTU on riittävän suuri. Toinen syy on tietoturvaan liittyvä. Suurella alustuspaketilla vältetään yhteyden ulkopuolisen henkilön sotkeutumista alustukseen ja pienennetään vahvistinhyökkäyksen (amplification attack) riskiä.

Palvelin vastaa asiakkaan alustavaan pakettiin omalla alustavalla paketillaan. Tämä pitää sisällään kuten asiakkaankin paketissa TLS:n ServerHello-viestin. Palvelin on nyt saanut riittävät tiedot avaintenvaihdossa TLS:n viestien avulla toteuttaakseen loppuihin yhteyteen liittyviin tietosähkeisiin salauksen. Asiakas saa muodostettua salausavaimen paketin salaamattomasta alusta.

Palvelimen vastaukseen sisältyy myös sen kättelypaketti (handshake packet). Tämän tehtävänä on kuljettaa vielä TLS1.3:n mahdollisia lisäparametreja kuten palvelimen varmenteen tai useita varmenteita. Tämän jälkeen palvelin lähettää heti perään vielä toisen kättelypaketin erillisessä UDP-tietosähkeessä. Tämä paketti sisältää vielä lisää salaustietoja, kuten sertifikaatin varmennuksen ja TLS:n kättelyn lopetusviestin.

Kun asiakas on vastaanottanut nämä kaksi pakettia palvelimelta, lähettää se vielä itse yhden UDP-tietosähkeen. Ensimmäinen paketti on jälleen alustuspaketin muotoinen, jossa kuitataan (acknowledge) palvelimen ensimmäinen paketti eli alustuspaketti. Kuten alkuperäisessä asiakkaan lähettämässä paketissa, pitää tähänkin lisätä nollia perään, jotta sen kooksi tulee yli 1200 tavua. Tämä johtuu siitä, että mukana on alustuspaketti. Samassa tietosähkeessä kulkee vielä kättelypaketti, joka kuittaa palvelimen lähettämän kättelypaketin. Tämän jälkeen yhteys on muodostettu ja tiedonsiirto voidaan aloittaa asiakkaan ja palvelimen välillä. QUIC-kättely on esitetty kuvassa 3.11.



Kuva 3.11. *QUIC-kättely. Muokattu lähteestä [8].*

Kättelyn jälkeen paketteja voidaan alkaa lähettää. Ensimmäiseen varsinaiseen datansiirtoon käytettyyn UDP-tietosähkeeseen liittyy vielä TLS-kättelyn viimeistely. Tässä lähetetään UDP-tietosähkeen mukana kättelypaketti ja sovelluspaketti (application packet). Kättelypaketti sisältää TLS-kättelyn viimeistelyviestin. Sovelluspaketti toimii QUICissa tiedonsiirtoa toteuttavana pakettina.

QUICissa pitää kättelypaketti kuitata aina erikseen, joten palvelimen pitää vielä seuraavassa UDP-tietosähkeessään kuitata kättelypaketti omalla ACK-kättelypaketillaan ja sen jälkeen lähettää oman sovelluspakettinsa vastauksena asiakkaan pyyntöön. Kun tämä viimeinen kättelypaketti on saatu hoidettua, voidaan dataa siirtää sovelluspaketeilla niin kauan kuin tarvitsee. Asiakas kuittaa aina jokaisen vastaanotetun sovelluspaketin erikseen palvelimelle.

Kun palvelin on saanut kuittaukset asiakkaalta kaikkiin lähettämiinsä paketteihin, on yhteys valmis suljettavaksi. Sulkeminen tapahtuu sovelluspaketilla, johon liitetään kehys, joka ilmoittaa yhteyden sulkemisesta. Tätä ei tarvitse enää kuitata vaan yhteys loppuu tämän paketin lähettämisen jälkeen. [5]

Kuvassa 3.11 on esitetty 1-RTT-yhteydenavaus (one round trip). Tämä tarkoittaa, että yhteyden alustamiseen ennen hyötykuorman lähettämistä menee yksi kokonainen verkkokierros eli edestakainen matka asiakkaalta palvelimelle. QUIC tukee myös yhteydenavausta 0-RTT:nä, mikäli asiakkaalla ja palvelimella on ollut aikaisempi yhteys vain vähän aikaa sitten. Tämä on tarkalleen ottaen TLS1.3:n ominaisuus, ja se on määritelty sen RFC 8446:ssa [23]. 0-RTT-yhteydenavauksessa ensimmäinen hyötykuormaosuus lähetetään heti ensimmäisessä UDP-tietosähkeessä. [17]

QUIC-yhteyden luonti saattaa epäonnistua erinäisistä syistä. Esimerkiksi alustuspaketti saattaa vaurioitua yhteydenavauksessa. Tällöin QUIC yksinkertaisesti lopettaa yhteyden muodostamisen. Mikäli virhe havaitaan ja QUIC-yhteyden muodostus lopetetaan, otetaan käyttöön TCP-yhteys saman datan siirtoon varaoptiona, sillä sen pitäisi olla aina käytettävissä eri laitteissa. QUIC siis tietyllä tapaa luottaa TCP:n saatavuuteen virheenhallinnassaan. [8]

Uutuudestaan huolimatta QUICia käytetään jo joillain verkkosivuilla. Googlen sivustot ovat näistä hyvä ja tunnettu esimerkki, sillä Google oli myös aktiivisesti mukana kehittämässä QUICia. W3Techs-sivuston statistiikan mukaan tällä hetkellä 8,5 prosenttia verkkosivuista käyttää QUICia. [25]

4. LUOTETTAVAN TIEDONSIIRRON PROTOKOLLIEN VERTAILU

Tässä työssä esiteltävistä protokollista TCP ja QUIC jakavat melko hyvin yhteisen käyttökohteen, internetin tiedonsiirron HTTP:llä. Hyvänä esimerkkinä tästä on luvussa 3.3 mainittu HTTP/3 (QUIC) vastaan HTTP/1.1 ja HTTP/2 (TCP). SCTP taasen toimii lähtökohtaisesti erilaisissa käyttötarkoituksissa ja sovelluksissa kuten signalointipalveluissa mobiiliverkoissa.

4.1 TCP ja SCTP

TCP:llä ja SCTP:llä on jonkin verran yhteistä, mutta erojakin niistä löytyy. Suurin eroavaisuus löytyykin käyttötarkoituksesta, jota jo sivuttiin yllä. SCTP on täysin kykenevä palvelemaan TCP:n roolissa internetin tiedonsiirrossa, mutta koska TCP on huomattavasti SCTP:tä vanhempi, on se myös vakiintuneempi ja laajemmin tuettu. Suuri ongelma on verkon välilaitteissa, kuten reitittimissä, joihin ei ole rakennettu sisään SCTP-tukea, ja niiden päivittäminen on kallista ja aikaa vievää. Nämä verkkolaitteet tukevat kuitenkin TCP/IP-mallin oleellisimpia protokollia, kuten TCP:tä ja UDP:tä, joten päivittämistä ei ole myöskään nähty tarpeelliseksi. Muun muassa tämän vuoksi SCTP ei ole merkittävä TCP:n kilpailija.

Koska SCTP:llä on aina assosiaatio luotuna, mahdollistaa se useamman yhtäaikaisen vuon luonnin päätepisteiden välille. Käytännössä tämä mahdollistaa sen, että eri paketit voidaan lähettää samanaikaisesti vastaanottajalle. TCP ei tätä suoraan tue. Tämä ei paranna tiedonsiirtonopeutta, mutta mikäli yksi paketti katoaa tai vaurioituu matkalla, se ei estä muiden pakettien lähetystä toisin kuin TCP:llä, joka lähettää paketin kerrallaan ja odottaa siitä kuittausta ennen seuraavan paketin lähetystä. Tämän myötä SCTP voi myös lähettää tietoa niin, että se ei välttämättä tule täysin alkuperäisessä järjestyksessä, kun taas TCP vaatii tarkan järjestyksen.

Toinen tiedonsiirron kannalta merkittävä ero on, miten nämä protokollat lähettävät tietoa. Käytännön erona TCP lähettää tiedot paketteina kun taas SCTP luo kehyksen, jonka puitteissa kaikki tieto lähetetään. Mikäli paketti on riittävän suuri eli yli MTU:n, pitää TCP:n pilkkoa siirrettävä tieto pienempiin osiin. Tämä aiheuttaa ylimääräistä tiedonsiirtoa, kun

TCP-pakettien otsikoiden koot huomioidaan kokonaistiedon määrään. SCTP-lähettää kehyksen, jonka jälkeen se voi lähettää koko siirrettävän tiedoston kerralla lohkoissaan, kuten kuvassa 3.5 on esitetty.

SCTP tuo mukanaan myös joitan turvallisuuskehityksiä TCP:hen verrattuna. Se on paremmin suojattu palvelunestohyökkäyksiä vastaan. TCP on herkkä niin kutsutulle SYN flood -hyökkäykselle, jossa palvelimelle lähetetään useita SYN-viestejä, jolloin se ei voi palvella oikeita asiakkaita. SCTP:n nelivaiheinen kättely on suunniteltu torjumaan tämäntyyppisiä hyökkäyksiä. [3]

Wen-Yuan, De-Wu ja Wei ovat vuoden 2009 käytännöllisessä tutkimuksessaan päässeet vastaaviin tuloksiin SCTP:n nopeudesta kuin aiemmin tässä luvussa on esitetty. Heidän tutkimuksensa paljasti, että voiden määrällä ei ole merkitystä tiedonlähetysnopeuteen. He myös huomasivat, että SCTP oli huomattavasti hitaampi protokolla verrattuna TCP:hen. Tähän syyksi esitettiin muun muassa SCTP:n optimoimattomuus käyttöjärjestelmissä verrattuna todella optimoituun TCP:hen. [21]

4.2 TCP ja QUIC

QUIC parantaa TCP:hen verrattuna verkkosivustojen tiedonsiirtoa. TCP on suunniteltu siirtämään luotettavasti yksittäisiä tiedostoja. Nykyajan verkkosivustot, jotka sisältävät muutakin sisältöä, kuten kuvia, tyylitiedostoja ja skriptejä, vaativat usean TCP-paketin siirtämisen. Tämä aiheuttaa viivettä ja hitautta, kun jokaiselle tiedostolle pitää luoda erillinen yhteys kättelyineen ja sulkemisineen. Dataa siis pitää siirtää huomattava määrä otsikoissa varsinaiseen hyötykuormaan verrattuna. QUIC pystyy kuitenkin lähettämään nämä paketit yhtäaikaisesti ja tehokkaammin kuin TCP. [8]

Kuten luvussa 3.3 todettiin, QUICin UDP-tietosähkeet sisältävät TLS:n kättelyn ja toteuttavat salauksen. TCP ei tällaista pakettien yhdistämistä tue, vaan TCP-kättelyn lisäksi salatun yhteyden saavuttamiseksi pitää tehdä vielä erillinen TLS-kättely luodun TCP-yhteyden sisällä. Eroavaisuus pinoissa on havaittavissa kuvasta 3.10. Tämä erillinen kättely lisää viivettä yhteyden muodostamisessa, mikä näkyy suoraan loppukäyttäjälle. Viive korostuu sitä enemmän, mitä kauempana asiakas ja palvelin toisistaan sijaitsevat. Toisin sanoen viive on molempien protokollien osalta sidottu verkkokierrosten (RTT) määrään. QUICissa RTT-määrä on 0 tai 1, kun taas TCP:ssä se on välillä 1-3.

QUIC vertautuu monella tapaa SCTP:hen, kun sitä vertaillaan TCP:hen. Edellisessä alaluvussa mainitut samanaikainen, epäjärjestyksellinen tiedonsiirto pätee QUICiin SCTP:n lailla, vaikka niiden toteutustavat hieman eroavatkin. Myös turvallisuusaspekti QUICissa on huomioitu, ja se käytännössä salaa kaiken lähettämänsä tiedon jo heti yhteyden alustuksen jälkeen. Myös hyvänä esimerkkinä suoraan protokollaa vastaan kohdistetun hyökkäyksen torjuntakeinosta on luvussa 3.3 manittu täytebittien käyttö alustuspaketeissa.

QUIC on huomattavasti TCP:tä tehokkaampi etenkin epävakaissa verkoissa. Tällaisissa verkoissa yhteydet ovat epävarmoja, joten TCP:n lähettämät paketit korruptoituvat tai katoavat herkästi matkalla, mikä aiheuttaa kaikkien pakettien lähetyksiin viivettä. QUIC pystyy hallitsemaan eri paketteja erillään toisistaan, joten vastaavaa viivettä ei havaita. Sama pätee SCTP:n yhteyksiin. [11]

Pitää huomioida, että vaikka QUIC tässä luvussa on esitetty huomattavasti TCP:tä paremmaksi protokollaksi, on sillä myös kääntöpuolensa. Yksi olennainen eriäväisyys on se, että QUIC on sovelluskerroksen protokolla. Tämä tarkoittaa, että sitä ei suoranaisesti ole toteutettu käyttöjärjestelmän ytimessä, vaan sen oikea toteuttaminen on eri sovellusten tehtävä. TCP:n voidaan olettaa toimivan lähes kaikissa verkkolaitteissa, joten se on ainakin tällä hetkellä varmemmin saatavissa oleva vaihtoehto.

APNIC (Asia-Pacific Network Information Centre) on kerännyt dataa QUICin käytöstä ja tehnyt siitä nopeusvertailuja ja julkaissut nämä blogissaan heinäkuussa 2022. Tulosten valossa QUICin päällä toimiva HTTP/3:n viive on useimmiten pienempi kuin TCP:n päällä toimiva HTTP/2. HTTP/3 toimittaa tiedot 2/3 kerroista nopeammin kuin HTTP/2. [2]

4.3 SCTP ja QUIC

Kuten kahdesta edellisestä alaluvusta huomattiin, SCTP ja QUIC ovat hyvin samankaltaisia protokollia, kun niitä verrataan TCP:hen. Molemmat luovat selkeän yhteyden tai assosiaation päätepisteiden välille, jota pitkin voidaan lähettää useita voita samanaikaisesti toisistaan riippumatta. Oleellinen kysymys siis on, miksi on käytetty aikaa ja resursseja uuden protokollan kehittämiseen.

Google on jo heti QUIC-protokollan alussa vuonna 2012 julkistanut dokumentin, missä perustellaan, miksi QUICia tarvitaan, ja SCTP ei ominaisuuksiltaan riitä. SCTP:n olemassaolo on siis tiedostettu ja sen puutteet on listattu dokumenttiin. Dokumentissa on myös määritelty useita reunaehtoja ja haluttuja ominaisuuksia uudelle protokollalle. Dokumentti kiinnittää erityistä huomiota SCTP:n viiveeseen. Tämä esiintyy SCTP:n kättelyssä ja virheenhallinnassa. Lisäksi huomiota on kiinnitetty kaistanleveyden epätehokkaaseen käyttöön salatuissa SCTP-yhteyksissä. Dokumentin mukaan suojatun SCTP-yhteyden luominen kestäisi 4 RTT:tä, joten QUIC on pääpiirteittäin pitänyt kiinni alkuperäisestä määritelmädokumentistaan ja vähentänyt viivettä huomattavasti SCTP:hen verrattuna. [14]

Luvussa 4.1 mainittu SCTP-tukemisongelma verkkolaitteissa on merkittävä syy QUICin kehittämiselle. Koska QUIC toimii UDP:n päällä, se ei kohtaa vastaavaa ongelmaa laitetuen kanssa, sillä UDP on laajasti tuettu verkkolaitteissa. Koska QUIC toimii sovelluskerroksella, sen tuki on myös helpommin lisätty päätelaitteille.

Pienempiä ominaisuudellisia eroja protokollista löytyy monta. QUIC-paketit ovat aina salattuja. SCTP:ssä pitää toteuttaa lisäominaisuuksia, jotka lisäävät viivettä salauksen to-

teuttamiseksi. QUIC on myös yleisesti ottaen kevyempi protokolla kuin SCTP, sillä monet sen ominaisuudet on toteutettu juuri sen käyttötarkoitusta eli HTTP/3:n tiedonsiirtoprotokollana toimimista varten. [1]

5. YHTEENVETO

Tämän työn tarkoituksena oli selvittää miten luotettavaa tiedonsiirtoa toteutetaan TCP/IP protokollapinon puitteissa. Tärkeänä tarkastelukohteena oli internetin tiedonsiirto. Vertailtaviksi protokolliksi valikoituivat TCP, SCTP ja QUIC, jotka kaikki pystyvät toteuttamaan luotettavaa tiedonsiirtoa, mutta toimivat hieman eri tavoilla. Näillä protokollilla on päällekkäisiä käyttötarkoituksia ja ominaisuuksia, joten ne sopivat hyvin esiteltäväksi yhdessä.

Protokollien perustoiminnallisuudet käytiin läpi perusteellisesti, jonka jälkeen näitä protokollia vielä vertailtiin toisiinsa. Vertailussa käytiin läpi protokollien eriäviä ominaisuuksia, mutta myös vertailtiin niiden suorituskykyä jo olemassaolevien tutkimuspaperien perusteella. Vertailussa kävi ilmi, että samankaltaisetkin tai samanlaiseen käyttötarkoitukseen tarkoitetut protokollat saattavat erota toisistaan merkittävästi, kun niitä tutkitaan tarkemmin. Työssä huomattiin myös, että vaikka protokollia on kehitetty, niitä ei välttämättä tueta kovinkaan hyvin verkon runkolaitteissa. Tämä on hyvä pitää mielessä tulevaisuuden protokollia suunnitellessa ja kehitettäessä.

Työhön otettiin mukaan QUIC-protokolla, sillä se on internetin mittakaavassa todella uusi protokolla. Tämä antoi hyvän näkökulman siihen, minkälaisia suunnitteluperusteita tulevaisuuden tiedonsiirtoprotokolliin sovelletaan. Viiveen minimointi tuli työssä tärkeäksi aiheeksi vertailussa, ja se on toiminut selkeänä suunnannäyttäjänä QUICin kehityksessä. Kun internet kasvaa ja yhä enemmän dataa siirretään ympäri maapalloa palvelinten sijaitessa milloin missäkin, alkaa viive luonnollisesti olemaan tärkeässä roolissa.

Koska päätelaitteiden ja loppukäyttäjien määrän kasvu internetissä ei ainakaan ole hidastumaan päin, voidaan miettiä, mitä tulevaisuus tuo tullessaan. QUIC näyttäisi ainakin tämän työn näkökulmasta olevan tulevaisuuden internetin tiedonsiirtoprotokolla, mutta sen yleistymistä pitää vielä hieman odottaa. Tulevaisuudessa saatetaan nähdä lisää QUICin kaltaisia protokollia, jotka toimivat eri käyttötarkoituksiin. Mahdollista on myös, että QUICia kehitetään eteenpäin tukemaan muutakin kuin pelkkää HTTP/3:a.

LÄHTEET

[1] *A Comparison between SCTP and QUIC*. Maaliskuu 2018. URL: https://datatracker.ietf.org/doc/html/draft-joseph-quic-comparison-quic-sctp-00#section-5 (viitattu 02. 12. 2022).

[2] *A look at QUIC use*. Heinäkuu 2022. URL: https://blog.apnic.net/2022/07/11/a-look-at-quic-use/ (viitattu 02. 12. 2022).

[3] *Difference Between SCTP vs TCP*. URL: https://www.educba.com/sctp-vs-tcp/ (viitattu 02. 12. 2022).

[4] *Difference Between Source Port and Destination Port*. URL: https://www.geeksforgeeks.org/difference-between-source-port-and-destination-port/ (viitattu 19. 10. 2022).

[5] Michael Driscoll. *The Illustrated QUIC Connection*. URL: https://quic.xargs.org/ (viitattu 30. 11. 2022).

[6] *How MTU and MSS Affect Your Network*. URL: https://networkdirection.net/articles/network-theory/mtu-and-mss/ (viitattu 20. 10. 2022).

[7] *HTTP3 over QUIC protocol*. Elokuu 2021. URL: https://docs.citrix.com/en-us/citrix-adc/current-release/system/http3-over-quic-protocol.html (viitattu 22. 11. 2022).

[8] Geoff Huston. *A quick look at QUIC*. Maaliskuu 2019. URL: https://blog.apnic.net/2019/03/04/a-quick-look-at-quic/ (viitattu 28. 11. 2022).

[9] *Internet*. URL: https://ourworldindata.org/internet (viitattu 26. 11. 2022).

[10] James F. Kurose ja Keith W. Ross. *Computer Networking, A Top-Down Approach, Sixth edition*. Pearson, 2013. 862 s.

[11] Omkar Nalawade, Amit Dhanwani ja Tejashree Prabhu. ”Comparison of Present-day Transport Layer network Protocols and Google’s QUIC”. Teoksessa: *2018 International Conference on Smart City and Emerging Technology (ICSCET)*. 2018, s. 1–8. DOI: 10.1109/ICSCET.2018.8537265.

[12] Evi Nemeth et al. *Unix and Linux System Administration Handbook*. Addison-Wesley, 2018. 1179 s.

[13] *QUIC, a multiplexed transport over UDP*. URL: https://www.chromium.org/quic/ (viitattu 14. 10. 2022).

[14] *QUIC: Design Document and Specification Rationale*. Joulukuu 2013. URL: https://docs.google.com/document/d/1RNHkx_VvKWyWg6Lr8SZ-saqsQx7rFV-ev2jRFUoVD34/edit?pli=1# (viitattu 02. 12. 2022).

[15] *RFC 768, User Datagram Protocol*. Elokuu 1980. URL: https://www.rfc-editor.org/rfc/rfc768 (viitattu 25. 11. 2022).

[16] *RFC 793: Transmission Control Protocol*. Syyskuu 1981. URL: https://www.rfc-editor.org/rfc/rfc793 (viitattu 30. 10. 2022).

[17] *RFC 9000: Stream Control Transmission Protocol*. Toukokuu 2021. URL: https://www.rfc-editor.org/rfc/rfc9000 (viitattu 15. 10. 2022).

[18] *RFC 9260: Stream Control Transmission Protocol*. Kesäkuu 2022. URL: https://www.rfc-editor.org/rfc/rfc9260 (viitattu 15. 10. 2022).

[19] *RFC 9293: Transmission Control Protocol (TCP)*. Elokuu 2022. URL: https://www.rfc-editor.org/rfc/rfc9293 (viitattu 15. 10. 2022).

[20] *SCTP association startup and shutdown*. Lokakuu 2022. URL: https://www.ibm.com/docs/en/aix/7.2?topic=protocol-sctp-association-startup-shutdown (viitattu 14. 11. 2022).

[21] Wen-Yuan Tan, De-Wu Xu ja Wei Chen. ”Research on the Transport Performance of SCTP”. Teoksessa: *2009 International Symposium on Computer Network and Multimedia Technology*. 2009, s. 1–3. DOI: 10.1109/CNMT.2009.5374598.

[22] *TCP Windowing: What Is It, Scaling, and Tips*. URL: https://www.extrahop.com/company/blog/2017/tcp-windowing/ (viitattu 19. 10. 2022).

[23] *The Transport Layer Security (TLS) Protocol Version 1.3*. Elokuu 2018. URL: https://www.rfc-editor.org/rfc/rfc8446 (viitattu 30. 11. 2022).

[24] *Transmission Control Protocol (TCP) Parameters*. URL: https://www.iana.org/assignments/tcp-parameters/tcp-parameters.xhtml (viitattu 19. 10. 2022).

[25] *Usage statistics of QUIC for websites*. URL: https://w3techs.com/technologies/details/ce-quic (viitattu 30. 11. 2022).

[26] David J. Wetherall ja Andrew S. Tanenbaum. *Computer Networks, Fifth Edition*. Pearson, 2010. 960 s.

[27] *What happens in a TLS handshake? | SSL handshake*. URL: https://www.cloudflare.com/learning/ssl/what-happens-in-a-tls-handshake/ (viitattu 20. 10. 2022).

[28] *What is User Datagram Protocol (UDP/IP)?* URL: https://www.cloudflare.com/learning/ddos/glossary/user-datagram-protocol-udp/ (viitattu 15. 10. 2022).

[29] Juha Vihervaara. *Computer Networking II*, s. 148. 159 s.

[30] Juha Vihervaara. *Internetin Tiedonsiirto, TCP/IP-tekniikan perusteet*, s. 74. 166 s.