Tampereen yliopisto

Ano Stén

HTTP/3-PROTOKOLLAN TOIMINTA JA SUORITUSKYKY

Kandidaatintyö
Informaatioteknologian ja viestinnän tiedekunta
Tarkastaja: Juha Vihervaara
Lokakuu 2023

TIIVISTELMÄ

Ano Stén: HTTP/3-Protokollan toiminta ja suorituskyky
Kandidaatintyö
Tampereen yliopisto
Tieto- ja sähkötekniikan tutkinto-ohjelma, Tietotekniikka
Lokakuu 2023

Hypertext Transfer Protocol 3 (HTTP3) on vuonna 1997 standardisoidun HTTP-protokollan uusin versio. HTTP perustuu asiakas-palvelin-arkkitehtuuriin, jossa palvelin vastaa asiakkaan pyyntöihin käytetyn metodin mukaisesti. Verkkoselaimet sekä verkkopalvelimet käyttävät HTTP:tä kommunikoidessaan keskenään. HTTP/3:ssa kuljetuskerroksen protokollana toimii edeltävistä versioista poiketen User Datagram Protocol (UDP) ja luotettava tiedonsiirto on toteutettu sen päällä toimivalla QUIC-protokollalla. Lisäksi Hyper Text Transfer Protocol Securen (HTTPS) mahdollistava Transport Layer Security -protokolla (TLS), joka edeltävissä versioissa toimi Transmission Control Protocolin (TCP) päällä, on HTTP/3:ssa sisällytetty QUIC:iin. HTTP/3-yhteydet ovat myös aina salattu TLS 1.3:lla tai sitä uudemmalla versiolla. Tämän työn tarkoitus on antaa lukijalle yleinen käsitys HTTP/3-protokollan ja sen protokollapinon toiminnasta sekä toteutuksesta.

Työ käsittelee HTTP/3:n kanssa yhdessä toimivat protokollat sekä aiempien versioiden toteutukset ja niiden ongelmat. Työ koostuu kolmesta osasta. Ensimmäiseksi käydään läpi vaadittavat taustatiedot HTTP/3:n ymmärtämiseen. Seuraavaksi käsitellään itse HTTP/3:a ja siihen liittyvän pinon protokollia tarkemmin. Lopuksi tarkastellaan HTTP/3:n suorituskykyä tutkivia julkaisuja ja vertaillaan niiden tuloksia.

HTTP/3 vastaa läheisesti toiminnaltaan HTTP/2:ta, mutta siinä on kaksi merkittävää parannusta: Nopeampi, jopa 0-round-trip time -yhteydenavaus (RTT) ja QUIC:in rinnakkaiset tiedonsiirtovirrat. QUIC:iin sisällytetty TLS mahdollistaa nopeamman kättelyn yhteyttä avatessa ja QUIC:in täysin rinnakkaisissa virroissa hävinneet paketit eivät viivästytä muita virtoja. HTTP/2:n TCP:llä toteutetuissa virroissa siltä ei voida välttyä. HTTP/3:n otsikot ovat pakattu HTTP/2:n kaltaisesti, mutta pakkaukseen on käytetty HPACK:in sijasta QPACK:iä, sillä QUIC ei takaa TCP:n kaltaisesti pakettien saapumista oikeassa järjestyksessä.

Tarkastelluissa tutkimuksissa havaittiin HTTP/3:n suoriutuvan HTTP/2:ta paremmin varsinkin mobiiliverkoissa. Kiinteän verkon tutkimuksissa tulokset olivat osittain ristiriitaisia, mutta tietyillä Quality of Experience -mittareilla (QoE) HTTP/3 oli myös kiinteässä verkossa HTTP/2:ta nopeampi. Tulokset vaikuttivat lupaavilta, mutta jatkotutkimukselle on aihetta.

Avainsanat: HTTP/3, QUIC, suorituskyky

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck –ohjelmalla.

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

ALPN	Application-Layer Protocol Negotiation
CSS	Cascading Style Sheets
DASH	Dynamic Adaptive Streaming over HTTP
DL	Downlink
EPC	Evolved Packet Core
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTP/1.1	Hypertext Transfer Protocol version 1.1
HTTP/2	Hypertext Transfer Protocol version 2
HTTP/3	Hypertext Transfer Protocol version 3
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
LCP	Largest Contentful Paint
LENA	LTE-EPC Network simulAtor
MAC	Message Authentication Code
NAT	Network address translation
P2P	Point-to-point
QoE	Quality of Experience
RFC	Request For Comments
RTT	Round-trip time
S/P-GW	Serving & Packet Gateway
SI	Speed Index
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UE	User equipment
URL	Uniform Resource Locator
eNB	E-UTRAN Node B
ns	Network Simulator

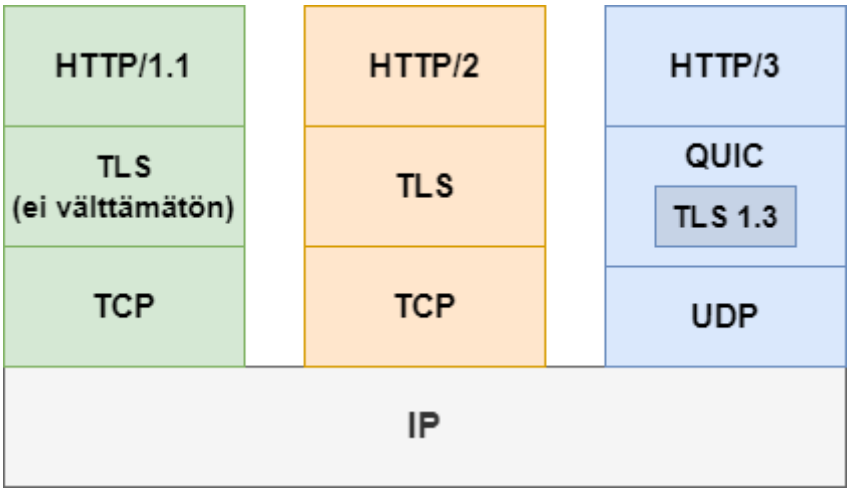
1. JOHDANTO

Hypertext Transfer Protocol (HTTP) on sovelluskerroksen protokolla, jonka ensimmäinen standardisoitu versio 1.1 julkaistiin vuonna 1997 [3]. HTTP perustuu asiakas-palvelin-arkkitehtuuriin, jossa palvelin vastaa asiakkaan pyyntöihin käytetyn metodin mukaisesti. Tavallisesti näistä pyynnöistä jokainen käsitellään toisistaan erillisinä tapahtumina, jolloin HTTP on siis tilaton protokolla. HTTP on välttämätön protokolla internetin selaamiseen, sillä verkkoselaimet ja verkkopalvelimet käyttävät sitä kommunikoidessaan keskenään. HTTP/3:a edeltävissä versioissa kuljetuskerroksen protokollana toimii Transmission Control Protocol (TCP), joka takaa HTTP:lle luotettavan tiedonsiirron. [16]

HTTP:n toinen versio HTTP/2 standardisoitiin vuonna 2014. Uuden version tuomia päivityksiä olivat verkkoresurssien käytön optimointi ja käyttäjän havaitseman viiveen vähennys käyttämällä pakattuja otsikkokenttiä. Toinen versio ei mitätöinyt versiota 1.1, sillä se mahdollisti vanhan syntaksin käytön. [1]

HTTP/3 on nimensä mukaan HTTP-protokollan kolmas versio, joka on tällä hetkellä standardisointivaiheessa. Edellisistä versioista poiketen HTTP/3 käyttää verkkokerroksen protokollana User Datagram Protocollaa (UDP) ja sen päälle rakennettua QUIC-protokollaa [2][7]. HTTP/3 kehitettiin ratkaisemaan edeltävien versioiden TCP pohjaisesta toteutuksesta johtuvia ongelmia, joita ei voitu ratkaista ainoastaan ylemmillä sovellustason ratkaisuilla, vaan tarvittiin uusi UDP:tä kuljetuskerroksella hyödyntävä protokolla. Kuva 1 esittää HTTP:n versioiden protokollapinoja.

Tämän työn tarkoitus on antaa lukijalle yleinen käsitys HTTP/3-protokollan ja sen protokollapinon toiminnasta sekä toteutuksesta. Työ käsittelee HTTP/3:n kanssa yhdessä toimivat protokollat sekä aiempien versioiden toteutukset ja niiden ongelmat. Kappale 2 tarjoaa lukijalle vaadittavan taustatiedon HTTP/3:n ymmärtämiseen. Kappale 3 koskee itse HTTP/3:a ja siihen liittyvän pinon protokollia tarkemmin. Kappaleessa 4 tarkastellaan HTTP/3:n suorituskykyä tutkivia julkaisuja ja vertaillaan niiden tuloksia.



Kuva 1. *HTTP:n versioiden protokollapinot. [4]*

2. HTTP/3:N TAUSTA

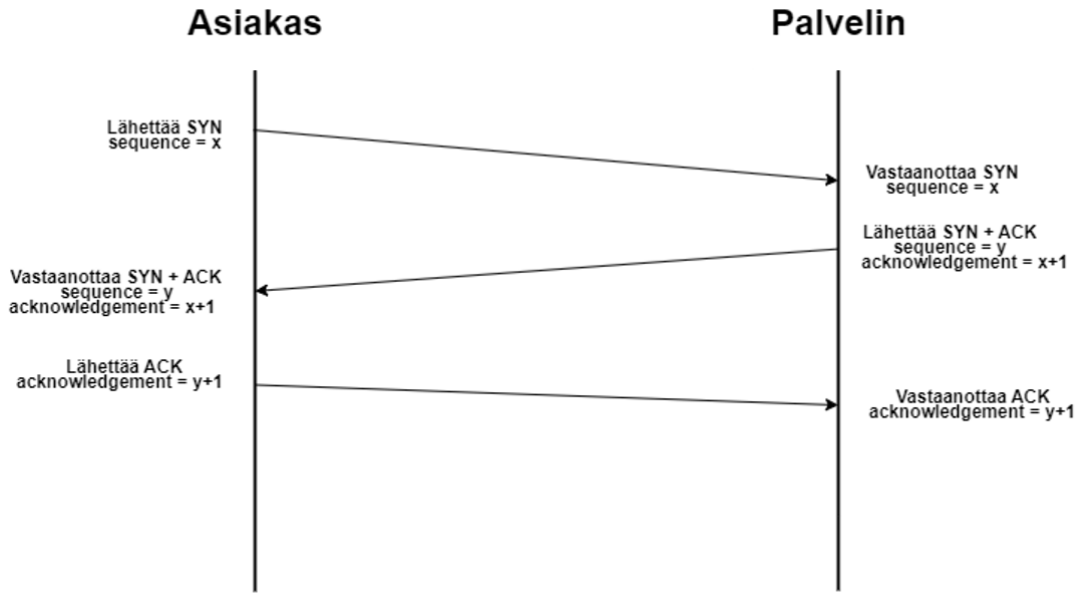
Tässä luvussa käsitellään TCP/IP-protokollapino sekä HTTP:n aikaisemmat versiot 1.1 ja 2, jotta voidaan tarkemmin ymmärtää HTTP/3:n toimintaa sekä sen kehitykseen johtaneita syitä.

2.1 TCP/IP

Laitteiden välinen viestintä verkossa perustuu TCP/IP-protokollapinoon, joka on jaettu neljään kerrokseen alhaalta ylös: peruskerrokseen, verkkokerrokseen, kuljetuskerrokseen ja sovelluskerrokseen. internetin kautta tapahtuvan viestinnän kannalta merkittävimpiä ovat verkkokerros ja kuljetuskerros. [10]

Verkkokerroksella toimiva Internet Protocol (IP) mahdollistaa pakettien kuljetuksen verkossa laitteiden välillä käyttäen laitteille annettuja IP-osoitteita. IP-osoitteiden avulla voidaan muodostaa verkkoja ja aliverkkoja. IP-osoitteet myös mahdollistavat pakettien reitityksen verkossa.

Kuljetuskerroksella toimii kaksi yleistä protokollaa, joita sovellustason protokollat käyttävät tiedonsiirtoon. Nämä protokollat ovat UDP ja TCP. TCP tarjoaa sovelluskerroksen protokollalle luotettavaa tiedonsiirtoa avaamalla laitteiden välille full duplex -tiedonsiirtoyhteyden kolmivaiheisella kättelyllä [10].



Kuva 2. *TCP:n yhteyden avauksessa tapahtuva kolmivaiheinen kättely. [6]*

Kuva 2 esittää yhteyttä avatessa suoritettavaa kolmivaiheista kättelyä, joka voidaan kuvata seuraavasti:

1. Asiakas lähettää palvelimelle paketin SYN-lipulla ja asettaa sequence-kenttään oman sekvenssinumeronsa x.
2. Palvelin vastaa paketilla, jossa SYN- ja ACK-liput ovat asetettuina, sekä asettaa sequence-kenttään oman sekvenssinumeronsa y ja acknowledgment number -kenttään arvon x+1.
3. Asiakas vastaa paketin vastaanotettuaan ACK-lipulla ja asettaa acknowledgement number -kenttään y+1.

Kolmannessa vaiheessa asiakas voi jo halutessaan lähettää hyötydataa ACK-viestin lisäksi. [6]

Vaihdettuja sekvenssinumeroita TCP käyttää lähetettyjen ja vastaanotettujen pakettien merkitsemiseen. Kolmivaiheisen kättelyn ohessa mainittu acknowledgement number -kenttä osoittaa laitteen seuraavaksi odottamaan pakettiin [10]. TCP siis takaa luotettavan tiedonsiirron uudelleenlähetyksillä, jotka perustuvat havaittuihin kuittausnumeroihin ja Retransmission Timeout ajastimeen [6].

UDP on huomattavasti TCP:tä yksinkertaisempi protokolla, sillä se ei takaa luotettavaa tiedonsiirtoa. UDP-otsikossa on vain neljä kenttää, jotka ovat lähdeportti, kohdeportti, segment length ja checksum [16]. Yksinkertaisuudessaan UDP vain lisää porttinumerot IP-osoitteiden pariksi. UDP:ta käyttäessä on mahdollista taata luotettava tiedonsiirto käyttämällä sen päällä erillistä luotettavaa tiedonsiirtoa tarjoavaa protokollaa [10].

2.2 HTTP/1.1

HTTP/1.1 on tekstipohjainen protokolla, johon kuuluu kahdenlaisia viestejä, jotka ovat pyyntö (request) ja vastaus (response). HTTP-tapahtuman alussa asiakas lähettää palvelimelle pyynnön, johon sisältyy metodi, joka kertoo palvelimelle mitä tehdä, sekä Uniform Resource Locator (URL), joka osoittaa kohteen. [16]

Lista HTTP/1.1:n metodeista löytyy protokollaan liittyvästä Request For Comment:sta (RFC). Niitä on esimerkiksi GET, joka pyytää palvelinta vastaamaan URL:n osoittamalla resurssilla [3]. Kaikki pyynnöt eivät palauta asiakkaalle resurssia. Esimerkiksi DELETE poistaa URL:n osoittaman resurssin palvelimelta.

Palvelimen vastauksessa pyyntöön on aina Status-Line-kentässä kolminumeroinen koodi, joka kertoo, miten palvelin vastasi pyyntöön. Koodeja on paljon, mutta ne ovat jaettu kategorioihin ensimmäisen numeron mukaan yhdestä viiteen. Täysi lista koodeista on löydettävissä RFC:ssä.

- Numerolla 1 alkavat ovat informatiivisia ja kertovat asiakkaalle, että pyyntö on vastaanotettu ja sitä käsitellään.
- Numerolla 2 alkavat kertovat pyynnön onnistuneen ja palauttavat entiteetin asiakkaalle viestin yhteydessä.
- Numerolla 3 alkavat kertovat uudelleenohjauksesta, jolloin asiakkaalta vaaditaan lisätoimia.
- Numerolla 4 alkavat ovat asiakasvirhekoodeja, jolloin kyseessä voi olla virheellinen syntaksi tai pyyntöä ei voida suorittaa.
- Numerolla 5 alkavat kertovat palvelinvirheestä, jolloin pyynnössä ei ollut vikaa, mutta palvelin ei muusta syystä voinut suorittaa pyyntöä. [16]

Pyyntö- ja vastausviestit eroavat rakenteeltaan toisistaan Kuva 3:n mukaisesti. Viestin ensimmäinen rivi on pyynnössä Request-Line ja vastauksessa Status-Line. Request-Lineen sisältyy metodi, URL ja HTTP-protokollan version. Esimerkiksi Kuva 3:n GET /kuva.jpg HTTP/1.1. Status-Linellä on HTTP-protokollan versio, tilannekoodi ja lyhyt ilmaus koodin merkityksestä. Status-Line aiemman esimerkkipyynnön onnistuessa on: HTTP/1.1 200 OK. Pyyntöviestissä kolmas kenttä on Request-Headers-kenttä ja vastausviestissä Response-Headers-kenttä. Request-Headers-kenttä sisältää erillisiä määritteitä, jotka loppuvat kaksoispisteeseen, ja niitä seuraa välilyönnillä erotettu arvo. Näitä ovat esimerkiksi ”Accept:”, joka listaa hyväksyttyjä mediatyyppejä ja ”Accept-

Language:”, joka kertoo toivotut kielet pyydetylle sisällölle. Vastausviestissä tämä kenttä tarjoaa lisätietoa vastaukseen liittyen. Esimerkiksi määrite ”Server:”, kertoo pyynnön käsitelleen ohjelmiston. [16]

Kuva 3. *HTTP pyyntö- ja vastausformaatti. [16]*

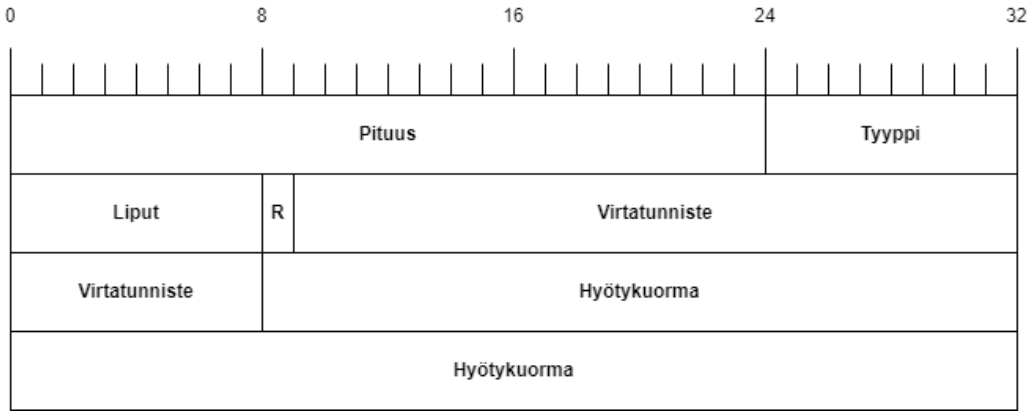
Molemmille yhteisiä kenttiä ovat General-Headers, Entity-Headers ja Entity-Body. General-Headers-kentässä on määritteitä, kuten ”Date:”, ”Connection:” ja ”Keep-Alive:”. Entity-Headers-kentän määritteet kertovat Entity-Body:n sisällöstä tai metodin kohteena olleesta palvelimen resurssista. Määritteistä ”Content-Type:” kertoo sisällön tyypin, ”Content-Length:” sisällön pituuden tavuina ja ”Last-Modified:” koska resurssia on palvelimen mukaan viimeksi muokattu. Entity-Bodyssä on pyydetty tai lähetetty data, jonka tulkinnassa Entity-Headers-kentän arvot auttavat, sillä Entity-Bodyssä oleva data voi olla eri tiedostomuodoissa esimerkiksi sovellusbitteinä tai eri standardeilla merkkikoodattua tekstiä. [16] Täysi lista HTTP/1.1:n määritteistä on löydettävissä RFC:stä [3].

Hypertext Transfer Protocol Secure (HTTPS) on yleisesti käytetty HTTP:n laajennos, jossa liikenne on salattu symmetrisesti käyttäen päällä Transport Layer Security (TLS) -protokollaa. HTTPS:ää käytettäessä on suoritettava yhteyttä muodostaessa TLS-kättely, joka versiosta päätellen hidastaa yhteyden avausta kahdella tai kolmella round-trip timellä (RTT). [14]

2.3 HTTP/2

HTTP/2:n suurin ero versioon 1.1 on kehysrakenne sekä otsikkokenttien pakkaus. Lisäksi HTTP/2 tarjoaa monta samanaikaista tiedonsiirtotapahtumaa samalla yhteydellä käyttäen virtoja. HTTP/2 tarjoaa lisäksi palvelimelle mahdollisuuden lähettää PUSH-tapahtumia asiakkaalle ilman asiakkaan pyyntöä sekä mahdollisuuden priorisoida valittuja pyyntöjä. [1] Vuonohjaus on myös mahdollista käyttämällä WINDOW_UPDATE-kehyksiä [11].

HTTP/2 ei ole HTTP/1.1:n kaltainen tekstipohjainen protokolla vaan se käyttää kehyksiä, jotka koostuvat biteistä. Kuva 4 esittää HTTP/2-kehystä, jonka tyyppi on esitetty type-kentässä. Kehystyyppejä on protokollassa kymmenen. HEADERS-tyypin kehys sisältää pakattuja otsikoita ja DATA-kehys virtaan liittyvää sisältöä [11]. Jos kaikki otsikot eivät mahdu yhden kehyksen sisään voidaan lisää otsikoita lähettää CONTINUATION-jatkokehyksessä. Jatkokehysten tarpeesta kertoo HEADERS-kehyksen otsikoiden loppua ilmoittava lippubitti [11].



Kuva 4. *HTTP/2-kehys. [11]*

Otsikoiden ollessa pakkaamattomia koostivat otsikot huomattavan osan lähetetystä ja vastaanotetusta datasta. Otsikoiden sisältö on myös toisteista, jolloin kuljetettiin paljon turhaa dataa. Turhan ja toisteisen datan vähentämiseksi otsikot pakataan HTTP/2:ssa käyttäen tehokasta HPACK-formaattia. HPACK käyttää staattisia ja dynaamisia tauluja, joilla toisteiset tai yleisessä käytössä olevat otsikot tai arvot ilmaistaan taulussa ilmoitetulla indeksillä [12]. Staattinen taulu sisältää kuusikymmentäyksi yleisintä otsikko-arvo-kombinaatiota ja on jokaisella laitteella sama [12]. Dynaaminen taulu kootaan yhteyden aikana, jolloin käytetyt otsikot ja arvot lisätään tauluun asetetun indeksoinnin mukaan joko yhdessä tai erikseen [11]. HTTP/2 on siis edeltäjästään poiketen osittain

tilallinen protokolla. HPACK pakkaa myös merkkijonot käyttäen Huffmannin koodausta [11]. Tehokkuuden maksimoinniksi kokonaisluvutkin pakataan [11].

HTTP/2 muodostaa vain yhden TCP-yhteyden palvelimeen mikä vähentää tarvittavia RTT:itä, sillä jokainen avattu yhteys vaatii oman TCP-kättelyn. HTTPS-tilanteessa olisi lisäksi suoritettava kullekkin yhteydelle TLS-kättely. Edeltävässä versiossa asiakkaan ja palvelimen välille saatettiin muodostaa maksimissaan kuusi yhteyttä, joista jokainen noudatti TCP:lle ominaista ruuhkan välttelyä ja hidasta aloitusta, joka ei verkkoresurssien kannalta ollut tehokasta. Virtojen avulla pyynnöt ja niitä koskevat vastaukset voidaan kuljettaa limittäin käyttäen yhtä TCP-yhteyttä. Uusia pyyntöjä voidaan siis lähettää, vaikka ensimmäiseen ei oltaisi saatu vielä vastausta. Virtoja voidaan avata lähettämällä HEADERS-kehys ja juokseva virtatunniste. [11] HTTP/2:n virrat eivät ole rinnakkaisia, vaan limittäisiä. Käytetty kuljetuskerroksen protokolla TCP takaa luotettavan tiedonsiirron, jossa paketit vastaanotetaan järjestyksessä. Kadonnut paketti yhdessä virrassa pysäyttää kaikki virrat kunnes kadonnut paketti on vastaanotettu.

Pyyntöjen tapahtuessa samanaikaisesti täytyy verkkosivun tärkeimpiä elementtejä priorisoida. Tämä ei ollut pakollista edeltävässä versiossa, sillä pyyntöjen käsittely järjestyksessä varmisti implisiittisesti tiedostojen latauksen oikeassa järjestyksessä. Selaimelle tärkeimmät elementit, kuten HTML (HyperText Markup Language), CSS (cascading style sheets) ja JavaScript, on ladattava ensimmäisenä, jotta selain voi alkaa jäsentämään verkkosivua mahdollisimman aikaisin. Priorisointi suoritetaan järjestämällä kahdella parametrilla: riippuvuuksilla ja painoilla. Riippuvuuksien avulla muodostetaan hierarkkinen tärkeysjärjestys ja painoilla tärkeysjärjestys hierarkiatasoilla. Asiakas ilmoittaa prioriteettinsa, mutta lopullinen päätösvalta niiden noudattamisesta jää palvelimelle. [11]

Palvelimien lähettäessä ennakoivia PUSH-tapahtumia käytettään PUSH_PROMISE-tyyppistä kehystä. Kehys sisältää sen virtatunnuksen jonka perusteella kehys lähetettiin, otsikon joka vastaisi asiakkaalle lähetettyyn resurssiin kohdistuvaa pyyntöä ja jonka metodi on idempotentti sekä virtatunnisteen jolla vastaus lähetetään. Sisältö lähetetään selaimen välimuistiin, ja sen lähetys voidaan aloittaa heti lupauksen lähetyksen jälkeen. Asiakas voi myös päättäessään hylätä PUSH:in. PUSH:ien tehokas toiminta on mahdollista vain, jos ennakointi toimii oikein ja sisältö lähetetään asiakkaalle nopeasti ennen asiakkaan omaa pyyntöä. [11]

3. HTTP/3:N TOIMINTA

HTTP/3 muistuttaa toiminnaltaan vahvasti HTTP/2:ta, ja sen tarkoitus on välttää edeltäjänsä kuljetuskerroksen TCP-protokollasta johtuvia rajoitteitta. Käyttämällä kuljetuskerroksella UDP:tä ja sen päällä uutta QUIC-protokollaa voidaan virrat erottaa toisistaan ja välttyä tilanteelta, jossa yksi virta viivästyttää muita. Näin saadaan useaa TCP-yhteyttä vastaava tilanne ilman TCP:n haittapuolia. [2] Tämä luku käsittelee HTTP/3-protokollan, sen alla toimivan QUIC-protokollan ja siihen liitetyn TLS 1.3 -protokollan sekä QPACK-pakkausformaatin.

3.1 HTTP/3

HTTP/3 sisältää HTTP/2:n ominaisuudet kuten: virrat, vuonohjaus, palvelimen PUSH, kehysrakenne ja otsikoiden pakkaus, mutta pakkausformaatti oli vaihdettava HPACK:istä uuteen QPACK:iin, sillä HPACK vaatii toimiakseen pakattujen kehysten vastaanottamisen oikeassa järjestyksessä, jota uusi QUIC-protokolla ei tarjoa. Sovellustason toiminnallisuuksia, joista vastasi aikaisemmin HTTP/2, on siirretty HTTP/3:ssa QUIC:ille. Esimerkiksi vuonohjauksesta ja rinnakkaisuudesta huolehtii nyt QUIC. [2]

HTTP/3:ssa on yksi- ja kaksisuuntaisia virtoja. Jokaiselle pyyntö-vastaus-tapahtumalle luodaan aina oma virtansa. Pyyntövirrat ovat kaksisuuntaisia virtoja, jotka kuljettavat pyyntöihin ja vastauksiin liittyviä kehyksiä. Lisäksi palvelimen lähettämät PUSH-lupaukset kulkevat pyyntövirroissa. Osapuolet sulkevat pyyntövirran aina omalta puoleltaan lähetettyään tapahtumaan liittyvät kehykset. Koska pyynnöille avataan aina omat virtansa, täytyy monen samanaikaisen virran olemassaolo olla mahdollista. RFC:ssä on annettu minimimääräksi sata rinnakkaista pyyntövirtaa. HTTP/2:ssa käytettiin lupausten yhdistämiseen virtatunnisteita, mutta HTTP/3:ssa käytössä ovat PUSH-tunnisteet, joilla yhdistetään yksisuuntaiset PUSH-sisältöä kuljettavat virrat vastaanotettuun lupaukseen. Myös uusien virtatyyppien lisääminen tulevaisuudessa on suunniteltu mahdolliseksi. [2]

Yksisuuntaisia virtoja käytetään PUSH:ien lisäksi QPACK:in enkooderi- ja dekooderivirtoina. Ehkäpä tärkein on ohjausvirta, ja niitä on yksi kummallakin päätteellä. Ohjausvirroissa kulkevat koko yhteyttä koskevat kehykset. PUSH-tunnisteiden maksimiarvo asetetaan MAX_PUSH_ID-kehyksessä, joka lähetetään ohjausvirrassa. Palvelin ei voi lähettää PUSH-lupauksia ennen kuin asiakas on lähettänyt

MAX_PUSH_ID-kehyksen. MAX_PUSH_ID:llä voidaan vain korottaa PUSH-tunnisteiden maksimiarvoa, mutta ei laskea sitä. Ohjausvirrassa lähetetään myös HTTP/3-yhteyden alussa SETTINGS-kehys, joissa osapuolet mainostavat itselleen sopivia arvoja toiselle osapuolelle. Näistä valitaan molemmille osapuolille yhteensopivat arvot, jotka koskevat koko yhteyttä eivätkä vain yksittäistä virtaa. Syynä yksisuuntaisten virtojen käyttöön on mahdollisuus pitää lähetysmahdollisuus aina avoinna kummassakin päässä. TLS mahdollistaa QUIC:in kanssa yhdessä 0-RTT -yhteydenavauksen. Jotta 0-RTT -yhteydenavausta voidaan käyttää, on laitteiden välillä täytynyt olla aikaisempi yhteys. 0-RTT -tapauksissa käytetään ensisijaisesti edellisen yhteyden asetuksia tai vaihtoehtoisesti vakioasetuksia, jos edelliset asetukset eivät ole enää QUIC-palvelimen muistissa. Tilanteessa, jossa asiakkaan käyttämät asetukset eivät ole yhteensopivia palvelimen muistissa olevien asetusten kanssa, 0-RTT -yhteydenavaus hylätään. [2]

HTTP/3-kehykset koostuvat kolmesta osasta, joista ensimmäinen on kokonaisluvulla ilmoitettu kehyksen tyyppi. Tyyppejä on yhteensä seitsemän ja lisäksi on vielä Reserved-tyypin kehykset. Reserved-kehyksiä voidaan käyttää sovellustason täytteenä, ja niitä voidaan lähettää jokaisen tyypin virrassa. HTTP/2:n kehyksiä vastaavat tyypit, joita ei ole HTTP/3:ssa, ovat Reserved-kehyksiä, mutta niiden lähettäminen johtaa virheeseen. Näitä ovat esimerkiksi HTTP/2:n vuonohjaukseen liittyvät kehykset. Toinen HTTP/3-kehyksen osa on pituus, joka ilmoittaa hyötykuorman pituuden tavuissa ja kolmas on hyötykuorma, jonka sisältö riippuu kehyksen tyypistä. Alla on listattu kaikki kehystyypit:

- DATA, joka sisältää pyyntöön tai vastaukseen liittyvää dataa.
- HEADERS, joka sisältää QPACK:illä pakattuja HTTP-otsikoita.
- CANCEL_PUSH, jolla PUSH-tunnisteen avulla estetään PUSH-virran avaaminen tai suljetaan jo avattu virta.
- SETTINGS, joka sisältää hyötykuormassaan nolla tai enemmän asetustunniste-arvo-paria.
- PUSH_PROMISE, joka sisältää luvatun QPACK:illä pakatun pyyntö-otsikon sekä PUSH-tunnisteen.
- GOAWAY, joka aloittaa HTTP/3 yhteyden hallitun katkaisun. GOAWAY-kehyksen lähetettyään lähettäjä lopettaa uusien pyyntöjen tai PUSH:ien vastaanottamisen. Kehyksen mukana lähetetään virtatunniste tai PUSH-tunniste. Tätä tunnistetta vastaava tapahtuma suoritetaan loppuun ennen yhteyden katkaisua.

- MAX_PUSH_ID, jolla asiakas määrittelee sallittujen PUSH-tapahtumien enimmäismäärän. [2]

HTTP/3:ssa on virhettä käsittelevien tilannekoodien lisäksi erillinen joukko virhekoodeja, joita lähetetään, kun virtoja tai koko yhteys on suljettava jostain syystä. Vastaavia virhekoodeja löytyy myös HTTP/2:sta. Ne ilmoittavat, että lähettäjä ei käsitellyt tai lähettänyt koko pyyntöä tai vastausta. Toisaalta tilannekoodit ovat vastaanotetun pyynnön tulos. Virhekoodi voidaan myös lähettää, kun turha virta suljetaan, jolloin mitään varsinaista ongelmaa ei tapahtunut. Koodia H3_NO_ERROR käytetään, kun halutaan sulkea virta, mutta ei ole viestittävää virhettä. H3_MISSING_SETTINGS lähetetään, kun yhteyttä avatessa ei lähetetty SETTINGS-kehystä ohjausvirrassa. Lisäksi on kehyksiä ja niiden sisältöä koskevat virheet esim. H3_MESSAGE_ERROR. [2]

HTTP/3:n kehykset ovat yksinkertaisia verrattuna HTTP/2:een, sillä QUIC:in ansiosta virtatunniste ja lippukenttä ovat tarpeettomia. Muita eroja HTTP/3:n ja HTTP/2:n välillä ovat esimerkiksi seuraavat. HTTP/3:ssa vuonohjaus koskee kaikkia kehyksiä, eikä vain DATA-kehyksiä. HTTP/3 ei myöskään tarjoa eksplisiittistä tapaa asettaa prioriteettiä, joka on HTTP/2:ssa mahdollista. PRIORITY-kehyksien lisäksi HTTP/3:ssa ei ole RST_STREAM-, PING- ja WINDOW_UPDATE-kehyksiä, vaan niitä vastaavista toiminnoista vastaa QUIC. CONTINUATION-kehystä ei myöskään ole, mutta vastineeksi HTTP/3 sallii suuremmat HEADERS- ja PUSH_PROMISE-kehykset. [2]

3.2 QUIC

QUIC on vuonna 2021 standardisoitu tilallinen sovellustason protokolla, joka hyödyntää UDP:tä. Se takaa luotettavan tiedonsiirron, ruuhkan hallinnan ja lisäksi viestien sisällön salauksen integroidulla TLS:llä. [7] Sovellustason protokollana QUIC:iin on myös helpompi tehdä päivityksiä, sillä se ei ole lukittu käyttöjärjestelmän kerneliin.

QUIC:in avaamat virrat merkitään kuusikymmentäkaksi-bittisellä, kokonaislukutunnisteella, joista parilliset ovat asiakkaan avaamia virtoja ja parittomat palvelimen. Toiseksi vähiten merkitsevä bitti kertoo, onko kyseessä yksi- vai kaksisuuntainen virta. Samoja tunnistenumeroita ei käytetä uudelleen saman yhteyden aikana. Tunnistenumeroita käytetään normaalisti juoksevasti, mutta tästä voidaan poiketa. Jos asetetaan tunnisteeksi järjestyksestä poikkeava numero, avataan myös muut pienemmän tunnistenumeron virrat, joilla on samat kaksi viimeistä bittiä. Näin pyritään takaamaan numeroinnin jatkuminen oikeassa järjestyksessä. [7]

Virrat siirtyvät eri tiloihin virran elinkaaren mukaan ja nämä poikkeavat toisistaan lähetys- ja vastaanottopuolilla. Tilojen vaihto perustuu lähetettyihin ja vastaanotettuihin

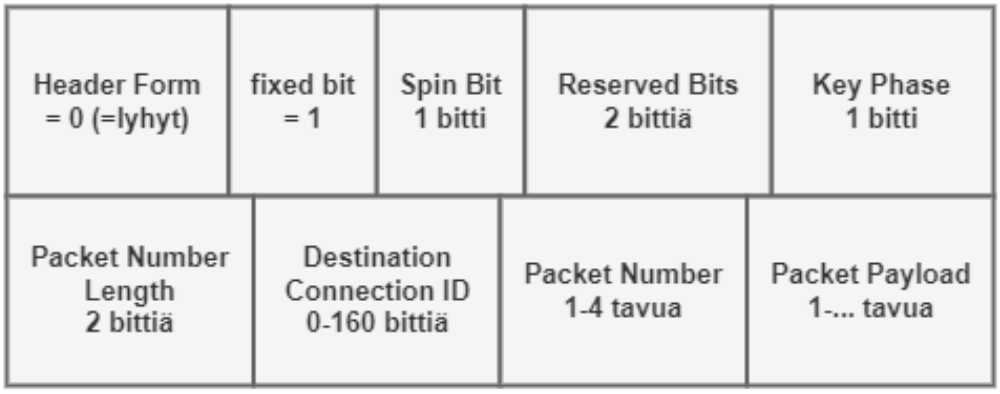
kehystyyppeihin. Data kulkee virroissa STREAM-kehyksissä, joissa on merkitty virran tunnus, sekä offset, jotta data voidaan erotella ja käsitellä oikeassa järjestyksessä. Tiloja ohjaavat kehykset ovat: RESET_STREAM, STREAM_DATA_BLOCKED, STOP_SENDING ja lipullinen STREAM. Sallittujen yhtaikaisten virtojen maksimimäärä päätetään yhteyttä avatessa ja määrää voidaan lisätä MAX_STREAMS-kehyksillä, mutta ei vähentää. Yksi- ja kaksisuuntaiset virrat rajoitetaan erikseen. [7]

Vuonohjausta virroissa suoritetaan vastaanottajapuolella MAX_STREAM_DATA kehyksillä, mutta niiden asettamat rajoitukset eivät koske uudelleenlähetyksiä. Vuonohjausta ohjataan myös koko yhteyden tasolla kehyksillä MAX_DATA. Näillä kehyksillä voidaan vain kasvattaa virtojen maksimaalista offsettiä tai kaikkien virtojen offsettien summaa. MAX_STREAM_DATA- ja MAX_DATA-kehykset, joissa arvo on edellistä pienempi, hylätään. Tilanteissa, joissa ilmoitettu maksimimäärä dataa on jo lähetetty, mutta osa on vielä lähettämättä, käytetään STREAM_DATA_BLOCKED ja DATA_BLOCK kehyksiä. Jos näitä kehyksiä ei lähetettäisi, voisi yhteys sulkeutua idle-ajan kuluttua. Kun käytetään MAX_STREAM_DATA JA MAX_DATA kehyksiä, on löydettävä tasapaino, kuinka paljon ikkunaa annetaan kerralla, jotta ei lähetetä liikaa kehyksiä suhteessa ikkunaan. Vuonohjauksen päivityksiä voidaan lähettää myös esimerkiksi ACK-kehysten mukana, joka vähentää turhaa verkon kuormitusta. [7]

Asiakkaan ja palvelimen välinen QUIC-yhteys avataan kättelyllä, joka poikkeaa Kuva 2:n TCP:n kättelystä. 0-RTT -tapauksessa asiakas voi lähettää sovellusdataa palvelimelle jo ennen vastausta palvelimelta. Palvelin voi myös lähettää sovellusdataa asiakkaalle ennen kättelyn salausosan viimeistä asiakkaan viestiä, jolla asiakas tunnistetaan. Palvelinpuolen ennenaikainen datan lähetys on riskialtista, mutta se myös vähentää viivettä. QUIC käyttää kahta eri tyyppistä pakettia kehysten lähettämiseen: pitkä- ja lyhytotsikkoisia paketteja. Pitkäotsikkoisia ovat Initial, 0-RTT, Handshake, Retry ja Version negotiation, ja niitä käytetään vain yhteyden muodostusvaiheessa. Kun yhteys on muodostettu ja 1-RTT -avaimet ovat saatavilla, siirrytään käyttämään lyhytotsikkoisia paketteja. Lyhytotsikkoisissa paketeissa kuljetetaan vähemmän turhaa dataa verrattuna pitkäotsikkoisiin, jolloin ne ovat verkkoresurssien kannalta tehokkaampia. Kuva 5 esittää lyhytotsikkoista QUIC-pakettia. [7] Kättely käsitellään tarkemmin TLS 1.3:n yhteydessä.

QUIC-yhteydet tunnistetaan yhteystunnisteella, jolloin yhteys ei ole riippuvainen alemman tason IP-osoitteista tai porteista. Molemmat päätteet käyttävät yhteydelle omaa tunnistettaan. Yhteystunnisteiden käyttö myös mahdollistaa connection migration -ominaisuuden tapauksissa, joissa asiakkaan IP-osoite tai portti muuttuu yhteyden aikana. [7] TCP:tä käyttäessä yhteys katkeaisi sillä yhteydet tunnistetaan IP-osoitteiden ja porttien kombinaatiolla [6]. Yhteys ei myöskään katkea, kun network address

translation (NAT) sidonnat muuttuvat. Tietoturvasyistä yhteydellä voi olla monta tunnistetta ja tunnisteisiin on liitetty sekvenssinumerointi, jotta tunnisteiden käsittelyä koskevat kehykset voidaan yhdistää tunnisteisiin. [7]



Kuva 5. *Lyhytotsikkoinen QUIC-paketti. [7]*

3.3 QPACK

QPACK vastaa erittäin läheisesti HPACK:iä sekä toiminnaltaan että toteutukseltaan. Myös QPACK, kuten HPACK, käyttää staattista sekä dynaamista taulua otsikoiden ja arvojen indeksointiin. Staattinen taulu on QPACK:issä pitempi ja koostuu yhdeksästäkymmenestäyhdeksästä kokonaislukuun indeksoidusta kombinaatiosta. Kuva 6 esittää QPACK:in staattisen taulun indeksit 14–24. Dynaaminen taulu muodostuu HPACK:in kaltaisesti QUIC-yhteyden aikana, mutta indeksointi on erillinen staattisesta taulusta. Indeksointeja on kolmea tyyppiä: Absolute, Relative ja Post-Base. Absoluuttinen indeksointi suoritetaan juoksevasti. Käyttämättömiä indeksejä voidaan poistaa taulusta, jos niihin ei olla hetkeen viitattu. Koska QUIC ei takaa vastaanotettujen pakettien saapuneen järjestyksessä, täytyy koodattujen otsikkojen mukana kuljettaa arvoa tauluun vaadituille lisäyksille. Tämä arvo on yhtä suurempi kuin käytettyjen viitteiden korkein indeksi. Jos dynaamista taulua ei ole käytetty paketin koodaukseen, on kentän arvo nolla. Kun dekooderi vastaanottaa koodatun otsikon, jonka vaadittujen lisäysten arvo on suurempi kuin sen oma lisäysten arvo, virta tukkiutuu ja tukoksen aiheuttanut otsikko säilytetään puskurissa. [8]

Enkooderi lisää viittaukset dynaamiseen tauluun ja huolehtii otsikkojen koodauksesta näihin indekseihin. Se myös pitää kirjaa tiedetyistä vastaanotettujen lisäysten määrästä. Tämän arvon avulla enkooderi tekee päätöksiä viittausten käytöstä välttääkseen virtojen tukkiutumisen toisessa päässä. Uudet viittaukset dynaamisessa taulussa lähetetään vastapuolelle enkooderivirrassa. Dekodeeri tulkitsee koodatut otsikot taulujen mukaan ja

pitää samalla kirjaa vaadittujen lisäysten ja tiedettyjen vastaanotettujen arvoista. Käsiteltyään dynaamiseen tauluun viittaavan otsikon lähettää dekooderi virrassaan kuittauksen. Datan pakkaamisen QPACK suorittaa samalla tavalla kuin HPACK. [8]

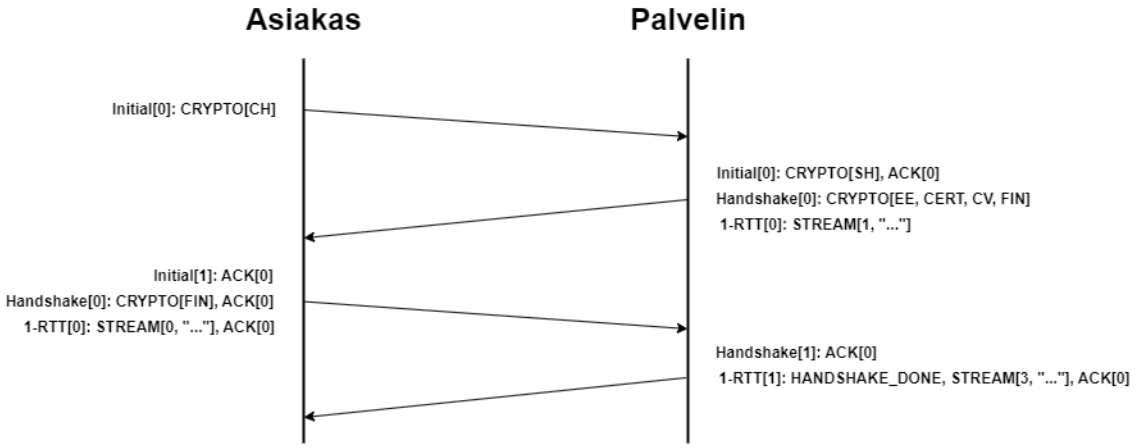
14	set-cookie	
15	:method	CONNECT
16	:method	DELETE
17	:method	GET
18	:method	HEAD
19	:method	OPTIONS
20	:method	POST
21	:method	PUT
22	:scheme	http
23	:scheme	https
24	:status	103

Kuva 6. *Osa QPACK:in staattisesta taulusta. [8]*

3.4 TLS 1.3

HTTP/3-yhteyksiä käytettäessä käytetään myös aina vähintään TLS versiota 1.3, josta käytetään tästä eteenpäin vain lyhennettä TLS. TLS käyttää datan salaukseen symmetristä salausta. Versiossa 1.3 merkittäviä muutoksia edelliseen ovat esimerkiksi karsittu määrä symmetrisiä salausformaatteja, 0-RTT -moodi sekä kättelyviestit ovat ServerHellon jälkeen salattuja. Kättelyvaiheessa on neljä suojaustasoa: Initial, 0-RTT, Handshake ja 1-RTT, joille luodaan avaimet ja vain 0-RTT tasolla ei kuljeteta CRYPTO-kehyksiä. Lähetetyt CRYPTO-kehykset salataan senhetkisellä salaustason avaimella, ja niiden sisällön avulla luodaan seuraavan tason avaimet. Koska dataa voidaan lähettää aikaisin ja hukkuneet paketit johtavat uudelleenlähetyksiin, on QUIC:in käytettävä samanaikaisesti monen salaustason avaimia. [17][13] QUIC:in normaali vuonohjaus ei koske TLS:ään liittyviä CRYPTO-kehyksiä, vaan niille on oma erillinen puskurinsa, jonka minimiarvoksi on asetettu 4096 tavua. [7]

Kuva 7 esittää 1-RTT -kättelyä uutta QUIC-yhteyttä avattaessa. Kuvatussa tapauksessa vain palvelimelta vaaditaan sertifikaattia. 0-RTT tapaus poikkeaa tästä kahdella tapaa. 0-RTT kättelyssä lähetetään ClientHellon lisäksi 0-RTT -dataa eikä palvelimen ensimmäisessä lähettämässä kättelypaketissa ole sertifikaattia tai sertifikaatin varmistavaa allekirjoitusta niiden ollessa tässä tapauksessa turhia. [7].



Kuva 7. *1-RTT -kättely. [7]*

Kättely jaettuna vaiheisiin:

1. Asiakas haluaa avata yhteyden ja lähettää palvelimelle Initial-paketissa CRYPTO-kehyksen sisällä ClientHello-viestin, joka sisältää asiakkaan tukemat versiot ja mahdollisen Application-Layer Protocol Negotiation (ALPN) extensionin.
2. Palvelin vastaa asiakkaalle omalla Initial-paketilla, joka sisältää CRYPTO- sekä ACK-kehykset. Initialin CRYPTO-kehys sisältää ServerHello-viestin, joka sisältää avain- ja salausasetukset. Palvelin lähettää myös Handshake-paketin, jonka sisällä on CRYPTO-kehyksessä EncryptedExtensions, sertifikaatti, sertifikaatin varmistus ja Finished. EncryptedExtensions sisältää vastauksia ClientHellossa lähetettyihin extensioneihin. Sertifikaatin varmistus sisältää sertifikaattiin yhdistetyllä yksityisellä avaimella salatun tiivisteen kättelyn aiemmista vaiheista. Salaus voidaan purkaa sertifikaatin julkisella avaimella ja varmistaa palvelin. Finished on Message Authentication Code (MAC), joka lasketaan edellisten viestien tiivisteen perusteella ja salataan base keyllä. Palvelin myös lähettää 1 RTT-paketissa STREAM-kehyksessä virran yksi sovellusdataa.
3. Asiakas vastaa Initial-paketilla, jossa on palvelimen Initial-pakettiin kohdistuva ACK-kehys. Lisäksi vastataan Handshake-paketilla, jossa on CRYPTO-kehyksessä Finished ja ACK-kehys, joka kohdistuu palvelimen lähettämään Handshake-pakettiin. Asiakas lähettää myös sovellusdataa 1-RTT -paketissa, jossa ovat STREAM-kehys ja ACK-kehys. STREAM sisältää virran 0 sovellusdataa. ACK kohdistuu palvelimen lähettämään sovellusdataan. Kättelyn voidaan sanoa olevan asiakkaan näkökulmasta nyt ohi.
4. Palvelin ilmoittaa asiakkaalle vastaanottaneensa Handshake-paketin lähettämällä siihen kohdistuvan ACK-kehyksen oman Handshake-pakettinsa sisällä. Tiedonsiirto jatkuu 1-RTT -paketeilla samanlaisesti kuin vaiheessa 3, mutta sisällä on lisäksi HANDSHAKE_DONE-kehys, joka ilmoittaa kättelyn päättyneen onnistuneesti.

Kättely on TCP:n kättelyä pitempi, mutta samalla suoritetaan TLS-kättely. Aiemmin täytyi HTTPS:ää käytettäessä suorittaa kättelyt erikseen, jolloin vaiheita on enemmän. Sovellusdatan lähetys voidaan myös aloittaa 1-RTT ja 0-RTT -tapauksissa aikaisemmin kuin TCP-yhteyttä avattaessa. [7][17][13]

4. HTTP/3:N SUORITUSKYKY

HTTP/3 ja sen alla toimiva QUIC-protokolla ovat suunniteltu toteutuksellaan vähentämään viivettä sekä toimimaan paremmin huonompilaatuisilla verkkoyhteyksillä, jossa pakettihäviöt ovat todennäköisempiä. Tässä luvussa tarkastellaan kolmea julkaistua tutkimusta sekä vertaillaan niiden tuloksia.

4.1 Ensikatsaus HTTP/3:n käyttöönottoon ja suorituskykyyn

Gianluca et al. suorittivat erittäin laajan tutkimuksen HTTP/3:n suorituskykyyn ja sen käyttöönottoon [4]. Tutkimuksessa vierailtiin yhteensä 14707 eri sivustolla samalla emuloiden muuttuvia verkko-olosuhteita: viivettä, pakettihäviöitä ja rajattua kaistanleveyttä. Kuukauden kestävän tutkimuksen aikana näillä sivustoilla käytiin 2 648 260 kertaa, jotta voitiin Quality of Experience (QoE) -mittareilla kartoittaa HTTP/3:n hyödyt selatessa internettiä.

HTTP/3 käyttöönottoa tutkittiin käyttämällä HTTPArchiven oikeilta käyttäjiltä Google Chrome -selaimen kautta saatua tietoaineistoa. Aineiston ollessa kokonaisuudessaan 6,6 TB karsittiin pois verkkosivut, jotka eivät olleet etusivuja. Näiden etusivujen HTTP-viesteistä etsittiin Alt-Svc-otsikoita, jotka ilmoittivat sivuston tukevan HTTP/3:a. Suorituskyvyn testaukseen kerättiin kolme tietoaineistoa, jotka koskivat eri HTTP/3:n käyttötapauksia: verkkosivujen käyttöä selaimella, mobiileissa verkko-olosuhteissa mobiililaitteella verkkosivujen käyttöä ja videon toistoa verkon yli.

Tietoaineisto kerättiin käyttäen automatisoitua BrowserTime työkalua, joka yhdisti automaattisesti verkkosivuille käyttämällä Google Chrome -selainta ja valittua HTTP-versiota. Verkko-olosuhteiden vaikutusta tutkittiin käyttämällä kahta palvelinta, jotka olivat yhdistetty internettiin Gigabit Ethernetillä. Verkko-olosuhteita simuloitiin Linuxin tc-työkalulla. Jokainen verkon konfiguraatio testattiin ensin pelkällä HTTP:n versiolla 1.1. Sitten versioilla 1.1 ja 2 sekä viimeiseksi kaikilla kolmella versiolla. Jokaisen testin välissä yhteyden tila ja välimuistit pyyhittiin. Testit toistettiin viisi kertaa tulosten luotettavuuden lisäämiseksi. Mobiililaitteita koskeva data kerättiin emuloimalla käyttäen Browsertimeä laiteen emuloinnissa ja ERRANT:ia mobiiliverkon emulointiin. Data kerättiin sadalta verkkosivulta käyttäen kuutta ERRANT-profiilia mobiiliverkon laadun hallintaan. Jokainen testi tehtiin kymmenen kertaa. Videon data kerättiin käyttäen Dynamic Adaptive Streaming over HTTP (DASH) web-clienttiä selaimessa, jossa toistettiin videoa HTTP-versioita 1.1, 2 ja 3 tukevasta palvelimesta. Video oli saatavilla kymmenellä eri

bittinopeudella. Videon toistoa testattiin kaikilla kolmella HTTP:n versiolla ja käyttäen kaikkia samoja verkon konfiguraatioita kuin aiemmin.

QoE-mittareista tutkittiin verkkosivuilla kahta: onLoad, joka vastaa verkkosivun täyden sisällön lataukseen ja käsittelyyn kulunutta aikaa sekä Speed Index (SI), joka vastaa verkkosivun näkyvien osien piirtymiseen ruudulle kulunutta aikaa. Videotesteissä tutkittiin videon resoluutiota, viivettä toiston alkuun ja resoluution muutoksia pienempään toiston aikana.

HTTP/3:a tukivat tutkimuksen mukaan 17,01 % verkkosivuista. Verkkosivutestauksen tulokset osoittivat HTTP/3:n suoriutuvan korkean viiveen verkko-olosuhteissa HTTP/2:ta paremmin. Kun viivettä lisättiin 50 ms, 70% verkkosivuista latautui QoE-mittareiden mukaan nopeammin HTTP/3:lla. Osuus kasvoi viiveen kasvaessa. Erittäin matalilla kaistanleveyksillä HTTP/3:n onLoad oli 57 %:lla verkkosivuista nopeampi. Pakettihäviötilanteissa ei huomattu eroja versioiden välillä. Verkkosivut, jotka hyötyivät eniten HTTP/3:a olivat ne, jotka rajoittivat yhteyksien ja kolmannen osapuolen domainien määrän, käyttivät HTTP/3:a koko sivulla ja olivat kooltaan pienempiä. Mobiiliverkon tuloksien mukaan 66–88 % verkkosivuista latautui HTTP/3:lla nopeammin riippuen käytetyistä verkko-olosuhteista ja laitteesta. Suurin etu HTTP/3:lle havaittiin hyvälaatuisessa 4G-verkossa. Mobiiliverkkojen viiveen havaittiin olevan 50–150 ms kokoluokkaa, joka tukee HTTP/3:n etuja korkean viiveen verkoissa. Videotestauksessa HTTP/3 suoriutui HTTP/2:ta huonommin matalilla kaistanleveyksillä ja sen aloitusviive oli korkeampi. Esitettyjä syitä videotestin tuloksille olivat QUIC:in ruuhkan hallinta, vuonohjaus ja niiden vaikutus soittimeen.

4.2 Varhainen Quality of Experience suorituskykymittaus HTTP/2:n ja HTTP/3:n välillä käyttäen Lighthousea

Saif et al. tekemässä tutkimuksessa vertailtiin QoE eroja HTTP/3:n ja HTTP/2:n välillä [15]. QoE mittaukset suoritettiin käyttäen avoimen lähdekoodin työkalua Lighthouse. Asiakkaana toimi Chrome Canary -selain ja palvelimen roolissa oli NGINX-palvelin, joka oli päivitetty tukemaan HTTP/3:a. Verkkoparametreja ohjattiin NetEm-nimisellä Linux-emulaatiotyökalulla. Vaihtelevat verkkoparametrit koskivat palvelimelta lähteviä paketteja. Palvelimelta pyydettävä verkkosisältö oli koostettu eri tiedostomuodoista. Tiedostojen koko, niiden sisältö ja kokosuhteet tiedostojen välillä perustuivat HTTPArchiven mediaaniverkkosivun sisältöön.

Käytetty Lighthousen versio oli 6.0.0. Lighthouse mittaa tarkastettavalta verkkosivulta piirteitä, jotka jaetaan viiteen kategoriaan. Mitatessa Lighthouse rajoittaa prosessorin ja verkon käyttöä tuloksia mitatessa, jotta otosdata voidaan normalisoida. Tutkimuksessa

keskityttiin vain tehokkuuskategoriaan. Tulokset pisteytetään kolmessa vaiheessa. Ensimmäisenä mitataan raakapisteet, joista lasketaan toisessa vaiheessa persentiili HTTPArchiven datan perusteella. Viimeisessä vaiheessa persentiilit lasketaan yhteen käyttäen Lighthousen eri osille asettamia painoja. Yhteenlasketut pisteet ovat välillä nollasta sataan. Kuva 8 sisältää käytetyt QoE-mittarit.

First Contentful Paint (FCP)	15 %	Aika, joka kuluu ensimmäisen Document Object Model -sisällön näkymiseen selaimessa, sivustolle saavuttaessa.
Time to Interactive (TTI)	15 %	1. FCP valmis 2. Sivustoelementtien käsittelijät latautuneet 3. Sivusto reagoi syötteeseen 50 ms ajassa.
Speed Index (SI)	15 %	Objektien ruudulle piirtymiseen kulunut aika.
Largest Contentful Paint (LCP)	25 %	Suurimman hyötykuorman elementin ruudulle piirtymiseen kulunut aika.
Total Blocking Time (TBT)	25 %	FCP:n ja TTI:n välissä tapahtuvat yli 50 ms tapahtumat lasketaan, ja kulunut 50 ms ylittävä aika summataan TBT:ksi.
Cumulative Layout Shift (CLS)	5 %	Kuinka paljon sivuston sisällön asetelma muuttuu sitä ladatessa. Suurempi pistemäärä tarkoittaa enempiä muutoksia asetelmassa.

Kuva 8. *Testeissä käytetyt QoE-mittarit. [15]*

Testit suoritettiin ensin ilman lisättyä viivettä HTTP/3:lle ja HTTP/2+TLS 1.3:lle. Palvelimelle oli testin aikana vain yksi yhteys, ja selaimen välimuisti tyhjennettiin testien välissä. Ensin tehtiin testit nousevilla viiveillä ja sitten nousevalla pakettihäviöllä käyttäen tasaista 300 ms viivettä. Jokainen testi toistettiin viisi kertaa, ja toistojen mittareista laskettiin keskiarvo. Mittarikeskiarvot muutettiin Lighthouse-pisteiksi siihen tarkoitetulla laskimella.

Tutkimuksen tuloksissa HTTP/3 suoriutui huonommin melkein kaikissa testeissä, mutta sen SI oli tasaisesti kilpailukykyinen. HTTP/3 oli aina hitaampi Largest Contentful Paintissa (LCP). Suoritusteho-testissä, jossa ladattiin 25 MB -kokoinen tiedosto, HTTP/3 voitti, mikä oli erikoista, sillä sen LCP oli aina hitaampi. Esitetty syy HTTP/3:n voittoon suoritustehossa oli tavallista aggressiivisemmaksi asetettu CUBIC. Korkean pakettihäviön tilanteessa suoriutui HTTP/3 paremmin, kun lähestyttiin testin sallittua 1,5 %:n pakettihäviön rajaa.

4.3 QUIC vs TCP: Suorituskyvyn arviointi LTE-verkossa käyttäen NS-3:a

Kyratzis et al. tutkivat QUIC:in suorituskykyä LTE-verkoissa käyttäen Network Simulatorin (ns) versiota 3.29. Vertauskohteena tutkimuksessa toimi TCP. [9] Verkon LTE-osat simuloitiin käyttämällä ns-3:n moduulia: LTE-EPC Network simulAtor (LENA), joka mallintaa todenmukaisesti LTE-protokollapinoa, sekä Evolved Packet Coren (EPC) palveluita. LENA:n käyttö rajoittaa simuloidun verkon käyttämään vain yhtä Serving & Packet Gatewaytä (S/P-GW). Simuloidussa verkossa oli yksi QUIC-palvelin ja yksi TCP-palvelin, kaksi User equipmenttiä (UE), S/P-GW ja E-UTRAN Node B (eNB). S/P-GW oli yhdistetty point-to-point-linkeillä (P2P) palvelimiin ja näillä linkeillä kapasiteetti oli 1 Gbps ja viive 12 ms. S1-U-linkillä kapasiteetti oli 1 Gbps ja viive 5 ms. UE:ista toinen käytti QUIC:ia ja toinen TCP:tä. Niiden liikenne oli reititetty vastaavalle palvelimelle.

Vakaan tilan suoritustehomittauksessa UE:t asetettiin yhtä kauas eNB:stä ja downlinkin (DL) kautta ajettiin UE:ille BulkSend-applikaatiolla jatkuvasti 512 tavua dataa. P2P linkkien pakettihäviöt asetettiin 0,5 %. Simulaatio kesti 40 sekuntia, joista ensimmäistä viittä ei huomioitu, jotta TCP:n hidas aloitus ei vaikuttaisi tuloksiin. Simulaatio toistettiin käyttäen eri etäisyyksiä. Vuon reiluutta simuloidessa mitattiin, kuinka reilusti QUIC jakaa käytettävän kaistan TCP:n kanssa LTE-verkossa. Simulaatiossa käytettiin BulkSendiä ja UE:n välimatka eNB:hen oli 250 m. Tiedoston latausta simuloitiin samalla etäisyydellä kuin edellisessä testissä. Simulaatiossa ladattiin viittä erikokoista tiedosta väliltä 64 kB–2 MB. Tiedoston lataukseen kulunut aika otettiin ylös. QUIC-virtojen vaikutusta suorituskykyyn testattin ns-3:n QUIC-applikaatiolla, joka lähettää QUIC-palvelimelta 300 µs välein 512 tavua dataa UE:lle. Simulaatio toistettiin käyttämällä vaihtelevaa määrää virtoja ja etäisyyttä.

Vakaan tilan tuloksissa QUIC oli nopeampi lyhyillä etäisyyksillä, mutta pidemmillä se hidastui TCP:n tasolle. Arveltu syy on heikommissa verkko-olosuhteissa pakettien uudelleenjärjestykseen käytetty aika. Reiluustestauksen tulokset osoittivat, että QUIC-yhteydet ovat reiluja toistensa suhteen, mutta eivät ole reiluja TCP:tä kohtaan, vaan varasivat simulaatiossa yli tuplasti TCP-yhteyden varaaman kaistan. Tiedoston lataus simulaatiossa osoitti QUIC:in lataavan jokaisen tiedoston nopeammin, ja että latausta voisi vielä nopeuttaa entisestään käyttämällä 0-RTT -yhteydenavausta. Viimeisessä simulaatiossa havaittiin, että suoritusteho kasvoi QUIC-virtojen määrän lisääntyessä, kun vertailukohtana olivat ensimmäisen simulaation tulokset. Toisaalta suoritusteho odotetusti väheni, kun UE loittoni gNB:stä. Virtojen lisäys helpottaa ongelmaa, jossa yhden paketin odotus viivästyttää muita.

4.4 Tulosvertailu

Gianluca et al. suorittaman tutkimuksen mukaan syyskuussa 2021 17,01 % verkkosivuista tukivat HTTP/3:a [4]. Huhtikuussa 2023 tuen prosenttiosuus on HTTPArchiven mukaan 18,80 % [5]. W3techsin mukaan osuus on 25,70 % [18]. Ero nykyhetken arvojen välillä on merkittävä ja poikkeukset aiheutuvat mahdollisesti laskutavasta. Kumpikin osuus on kuitenkin vuotta 2021 suurempi, joka voi osoittaa, että tuki on lisääntynyt.

Gianluca et al. saamien tulosten perusteella HTTP/3 suoriutuu HTTP/2:ta paremmin korkean viiveen verkoissa. Viiveen ollessa 50 ms 70 % verkkosivuista latautui nopeammin HTTP/3:lla, ja osuus kasvoi viiveen kasvaessa. Gianluca et al. suorittamissa mobiiliverkkotesteissä HTTP/3:a käyttäessä 66–88 % verkkosivuista latautui nopeammin vaihdellen käytetyn laitteen ja verkko-olosuhteiden mukaan. Mobiiliverkon viiveiden ollessa 50–150 ms kokoluokkaa suosivat mobiiliverkot viiveelle vastustuskykyistä HTTP/3:a. HTTP/3:n suorituskykyä puoltaa myös Kyratzis et al. tekemän tutkimuksen tulokset [9]. QUIC, jota HTTP/3 käyttää, suoriutui Kyratzis et al. suorittamissa testeissä tasaisesti TCP:tä paremmin mobiiliverkossa. QUIC latasi jokaisen testeissä ladatun tiedoston nopeammin kuin TCP, ja suoritustehon havaittiin kasvavan QUIC:in virtoja lisätessä. Virtojen lisääminen ei kuitenkaan nosta suoritustehoa loputtomasti, vaan virtojen lisäyksellä vähennetään hävinneiden pakettien vaikutusta muihin virtoihin. Kyratzis et al. mainitsivat myös, että 0-RTT -yhteydenavaus tiedoston latausta testatessa olisi kallistanut tuloksia vielä enemmän QUIC:in puolelle.

Saif et al. suorittaman tutkimuksen tulokset olivat ristiriidassa Gianluca et al. ja Kyratzis et al. tulosten kanssa [15]. Saif et al. tuloksissa HTTP/3 suoriutui tasaisesti HTTP/2:ta huonommin. SI, jota myös Gianluca et al. mittasivat, pysyi jokaisessa testissä kilpailukykyisenä, mutta muut käytetyt QoE-mittarit näyttivät HTTP/3:n suoriutuvan heikommin. Varsinkin LCP, joka oli jokaisessa testissä HTTP/3:lla hitaampi. Saif et al. huomasivat lisäksi HTTP/3:n suoriutuvan paremmin pakettihäviöiden kasvaessa, jota Gianluca et al. eivät havainneet. Gianluca et al. ja Saif et al. tuloksien eroa voivat selittää käytetyt QoE-mittarit. Jos Gianluca et al. olisivat mitanneet LCP:tä, olisivat tulokset voineet mahdollisesti poiketa nykyisistä.

5. YHTEENVETO

HTTP/3:n toiminta perustuu vahvasti sitä edeltävään HTTP/2:een, mutta kuljetuskerroksen protokolla on vaihdettu TCP:stä UDP:hen. UDP:n päällä toimii uusi QUIC-protokolla, joka takaa luotettavan tiedonsiirron ja jonka vastuulle on siirretty joitakin aikaisemmin HTTP/2:lle kuuluvia toimintoja, kuten virrat ja vuonohjaus. HTTP/3:ssa virrat ovat rinnakkaisia, eikä hävinnyt paketti odotuta muita virtoja. Käytetty pakkausformaatti on myös vaihdettu HPACK:istä QPACK:iin, sillä QUIC ei takaa pakettien saapumisjärjestystä. HPACK ja QPACK ovat toteutukseltaan muuten erittäin samanlaisia. QUIC:iin on sisällytetty lisäksi vähintään TLS-protokollan versio 1.3, ja HTTP/3 yhteydet ovat aina salattuja.

QUIC:in ja TLS:n käyttö yhdessä mahdollistaa HTTP/3:lle aikaisempaa nopeamman yhteyden avauksen. Aikaisemmissa versioissa suoritettiin ensin kolmivaiheinen kättely ja sitä seuraava TLS-kättely, joka vei kolmivaiheisen kättelyn lisäksi versioista riippuen kaksi tai kolme RTT:tä. Uudella kättelyllä voidaan lähettää hyötydataa 1-RTT -tapauksessa yhden RTT:n kuluessa. Tapauksessa, jossa asiakkaan ja palvelimen välillä on aikaisemmin ollut yhteys, voidaan käyttää 0-RTT -kättelyä, jolloin hyötydataa voidaan lähettää jo ensimmäisessä paketissa. Nopeamman kättelyn ja rinnakkaisen tiedonsiirron lisäksi connection migration -ominaisuuden avulla voidaan HTTP/3-yhteys pitää aktiivisena vaikka laitteiden porttinumerot tai IP-osoitteet muuttuisivat. Connection migration on mahdollista, sillä yhteydet erotellaan laitekohtaisilla yhteystunnisteilla. HTTP/3:n virrat mahdollistavat aiempaa paremman suorituskyvyn heikommissa verkko-olosuhteissa ja connection migration on selkeästi mobiileille laitteille suunniteltu ominaisuus, jossa laite liikkuu eri tukiasemien välillä.

HTTP/3 on suorituskyvyltään mobiiliverkoissa edeltäjäänsä tehokkaampi. Gianluca et al. tutkimuksessa havaittiin, että 66–88 % verkkosivuista latautui nopeammin HTTP/3:lla mobiiliverkoissa kuin HTTP/2:lla. Mobiiliverkoissa havaittiin olevan luontaisesti viivettä, jolle HTTP/3 on vastustuskykyinen. Myös kiinteässä 50 ms viiveellisessä verkossa HTTP/3 latasi 70 % verkkosivuista nopeammin kuin HTTP/2. Kyratzis et al. QUIC:ia ja TCP:tä vertailevan tutkimuksen tulokset myös osoittivat suoritustehon olevan korkeampi, tiedostojen latautuvan nopeammin ja QUIC:in virtojen nostavan suoritustehoa. Saif et al. tutkimuksessa, jossa vertailtiin HTTP/3:a edeltäjäänsä, havaittiin poikkeuksellisesti HTTP/3:n suoriutuvan QoE-mittareilla huonommin. Pakettihäviöllisessä verkossa Gianluca et al. tutkimuksesta poiketen HTTP/3 suoriutui paremmin pakettihäviöiden kasvaessa kohti testausympäristön 1,5 % rajaa.

HTTP/3:n nykyinen käyttöaste on noin 18,80 % HTTPArchiven mukaan, joten protokollan käytössä ei ole tapahtunut räjähdysmäistä kasvua syyskuusta 2021. On mahdollista, että käyttö kasvaa, kun HTTP/3 standardisoidaan virallisesti. Tutkimusten tulokset osoittavat HTTP/3:n sopivan hyvin käyttöön mobiiliverkoissa, mutta ne eivät ole kaikki toistensa kanssa yhtä mieltä. Epävarmuuden poistamiseksi olisi hyvä suorittaa Gianluca et al. tutkimuksen kaltainen laaja jatkotutkimus, joka mittaa QoE:ta käyttäen suurempaa määrää mittareita, jotta voidaan tarkastaa, ovatko Saif et al. tulokset uusittavissa. Mobiililaitteiden ollessa yleisiä, voivat HTTP/3:n mobiiliverkoissa edulliset piirteet johtaa sen lisääntyvään käyttöönottoon tulevaisuudessa.

LÄHTEET

[1] M. Belshe et al., Hypertext Transfer Protocol Version 2 (HTTP/2), May 2015. Saatavissa: https://datatracker.ietf.org/doc/html/rfc7540

[2] M. Bishop, Akamai, HTTP/3, June 2022. Saatavissa: https://datatracker.ietf.org/doc/html/rfc9114

[3] R. Fielding et al., Hypertext Transfer Protocol -- HTTP/1.1, June 1999. Saatavissa: https://datatracker.ietf.org/doc/html/rfc2616

[4] P. Gianluca et al., A First Look at HTTP/3 Adoption and Performance, Computer communications 187 (2022): 115–124.

[5] HTTP/3 Support, HTTPArchive. Saatavissa: https://httparchive.org/reports/state-of-the-web#h3 (Viitattu: 10.05.2023)

[6] Information Sciences Institute University of Southern California, TRANSMISSION CONTROL PROTOCOL, September 1981. Saatavissa: https://datatracker.ietf.org/doc/html/rfc793

[7] J. Iyengar et al., QUIC: A UDP-Based Multiplexed and Secure Transport, May 2021. Saatavissa: https://datatracker.ietf.org/doc/html/rfc9000

[8] C. Krasic, M. Bishop, A. Frindell, QPACK: Field Compression for HTTP/3, June 2022 (updated December 2022). Saatavissa: https://datatracker.ietf.org/doc/html/rfc9204

[9] A.I. Kyratzis, P.G. Cottis. QUIC vs TCP: A Performance Evaluation over LTE with NS-3. Communications and Network, 14, 12-22. 2022.

[10] T. Lammle, TCP/IP, Indianapolis, Indiana: Sybex, 2018, s. 13-15 ,26-29.

[11] S. Ludin, J. Garza. Learning HTTP/2 : a practical guide for beginners. First edition, Beijing, China: O'Reilly, 2017.

[12] R. Peon et al., HPACK: Header Compression for HTTP/2, May 2015. Saatavissa: https://datatracker.ietf.org/doc/html/rfc7541

[13] E. Rescorla, Mozilla, The Transport Layer Security (TLS) Protocol Version 1.3, August 2018. Saatavissa: https://datatracker.ietf.org/doc/html/rfc8446

[14] E. Rescorla, RTFM, HTTP Over TLS, May 2000. Saatavissa: https://datatracker.ietf.org/doc/html/rfc2818

[15] D. Saif, C. -H. Lung, A. Matrawy, An Early Benchmark of Quality of Experience Between HTTP/2 and HTTP/3 using Lighthouse, ICC 2021 - IEEE International Conference on Communications, Montreal, QC, Canada, 2021, s. 1-6.

[16] W. Stallings, MM. Manna, Data and computer communication. Tenth edition, Boston: Pearson, 2014, s. 66 ,826-837.

[17] M. Thomson et al., Using TLS to Secure QUIC, May 2021. Saatavissa: https://datatracker.ietf.org/doc/html/rfc9001

[18] Usage statistics of HTTP/3 for websites, W3Techs. Saatavissa: https://w3techs.com/technologies/details/ce-http3 (Viitattu: 10.5.2023)