Toni Kuikka

UNITY-PELIN ASSETBUNDLE JAKELUN OPTIMOINTI

Tradenomi

Kevät 2016

KAJAANIN
AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

KAJAANIN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

TIIVISTELMÄ

Tekijä(t): Kuikka Toni

Työn nimi: Unity-pelin AssetBundle jakelun optimointi

Tutkintonimike: Tradenomi, tietojenkäsittely

Asiasanat: Unity, Apache, Nginx, IIS, Välimuistitus, HTTP, CDN

Tämä opinnäytetyö toteutettiin selvitystyönä Critical Force Entertainmentille. Critical Force Entertainment on kajaanilainen pelialan yritys, joka kehittää työn kirjoitushetkellä Critical Ops –räiskintäpeliä. Työn tarkoitus oli selvittää, millaisilla ratkaisuilla pelin AssetBundle-tiedostojen jakelu verkon yli pelaajille olisi nopeinta ja luotettavinta. Ratkaisujen arvioinnissa otettiin huomioon myös kustannustehokkuus.

Ratkaisujen arvioinnin pohjaksi tutustuttiin moniin tekijöihin jotka vaikuttavat tiedostojakeluun verkon yli. Näitä tekijöitä ovat muun muassa käytetty web-palvelinsovellus, välimuistus, käytetty yhteysprotokolla ja CDN-palvelun hyödyntäminen. Tutustumisessa kerättiin tietoa eri tekijöiden vaikutuksen määrästä ja käytiin läpi vertailuja eri ratkaisujen välillä. Lisäksi kerättiin taustatietoa Unitysta ja AssetBundleista, jotta kyettiin paremmin arvioimaan eri ratkaisujen soveltuvuutta AssetBundle jakeluun. Pohjustuksen jälkeen arvioitiin eri tekijöiden soveltuvuutta AssetBundle jakeluun ja joidenkin tekijöiden vaikutuksen huomattavuutta testattiin käytännössä.

Työn lopputuloksena Nginx todettiin AssetBundle jakeluun sopivimmaksi web-palvelinsovelluksesi ja välimuistituksen hyödyntäminen erittäin kannattavaksi. Myös käytetyllä yhteysprotokollalla ja HTTP pakkauksella todettiin olevan huomattava vaikutus tiedostojakeluun verkon yli, mutta niiden todettiin olevan soveltumattomia AssetBundle jakeluun. Lisäksi CDN-palvelun hyödyntäminen todettiin kannattavaksi, mutta kustannussyistä vasta pelaajamäärien kasvettua hyvin suuriksi. Lopuksi todettiin, että luotettavimman tiedon eri tekijöiden vaikutuksesta AssetBundle jakeluun saisi testaamalla niitä käytännössä pelin todellisilla pelaajamäärillä mahdollisuuden ilmaantuessa.

KAJAANIN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

ABSTRACT

Author(s): Kuikka Toni

Title of the Publication: Optimizing the serving of Unity AssetBundles

Degree Title: Bachelor of Business, Computer science

Keywords: Unity, Apache, Nginx, IIS, caching, HTTP, CDN

This thesis was commissioned by Critical Force Entertainment, a video game company from Kajaani that is currently developing the shooter game Critical Ops. The purpose of the thesis was to find out how to optimize the game's AssetBundle delivery to be as fast and reliable as possible. Cost efficiency was also taken into consideration.

As a foundation for finding the best solutions for AssetBundle delivery optimization, different factors that affect file distribution over the internet were researched. Examples of these factors include the chosen web server, caching, the chosen communications protocol and the use of a CDN service. The effect of different factors was evaluated, and comparisons of different solutions were studied. Information of Unity and AssetBundles was also collected for the purpose of determining the applicability of the different factors for AssetBundle delivery. After this foundation the applicability of the different factors was evaluated and the effect of some of the factors was tested in practice.

For AssetBundle delivery Nginx was concluded the most suitable web server, and caching was found to be very beneficial. The HTTP/2 protocol and HTTP compression were also found to improve the speed of file distribution over the internet, but they were not applicable for AssetBundle delivery. With a large playerbase spread across many different continents, a CDN service should also be used. It was also concluded that for the most reliable information of the effect of different factors for AssetBundle delivery, the solutions should be tested in practice with real player numbers when possible.

SYMBOLILUETTELO

CDN	Content Delivery Network. Palvelu, jonka tarkoitus on nopeuttaa sisällön jakelua ja parantaa jakelun varmuutta.
Dynaaminen sisältö	Yksilöityjen vastausten jakelu palvelimelta sisältäen tietoa esimerkiksi tietokannasta. Esimerkiksi käyttäjän oma profiilisivu.
Pelimoottori	Ohjelmistokehys pelien kehitykseen.
PHP	Palvelimilla usein käytetty ohjelmointikieli
Protokolla	Yhteyskäytäntö laitteiden tai ohjelmien välisille yhteyksille.
SSL-kättely	SSL-salatun yhteyksen aloitukseen tarvittavat rutiinit palvelimen ja asiakasohjelman välillä.
SSL-salaus	Standardoitu salausteknologia palvelimen ja asiakasohjelman väliselle liikenteelle.
Staattinen sisältö	Tiedostot, jotka jaetaan palvelimelta sellaisenaan
Tiiviste	Pienempään tilaan tiivistetty muoto tiedosta. Tiivisteitä vertaamalla voidaan tarkistaa onko molempien tiivisteiden alkuperäinen tieto sama.
URL uudelleenkirjoitus	Web-palvelinsovellusten toiminto, jota käyttäen selkeät URL-osoitteet voidaan ohjata oikeisiin palvelimen resursseihin.
Välityspalvelin	Kahden eri kohteen, usein palvelimen ja asiakasohjelman, väliseen liikenteeseen määritetty välietappi.

Sisältö

1 JOHDANTO

Tämä opinnäytetyö tehtiin toimeksiantona Critical Force Entertainmentille. Critical Force Entertainment on kajaanilainen pelialan yritys, joka kehittää työn kirjoitushetkellä Critical Ops -räiskintäpeliä. Työn tarkoituksena on tutustua web-palvelinten ja web-jakelun optimointiin ja selvittää niiden sovellettavuutta Critical Ops -pelin AssetBundle-tiedostojen jakeluun. Tavoitteena on muodostaa ratkaisuja, joilla voidaan maksimoida AssetBundle jakelun nopeus ja palvelinten kyky palvella suuria käyttäjämääriä.

Työn tekohetkellä Critical Ops -pelin AssetBundle jakeluun käytetty ratkaisu on edullinen, mutta ei optimaalinen. Se kuitenkin toimii nopeasti ja ongelmitta pelin nykyisillä pelaajamäärillä. Pelaajamäärät kuitenkin kasvavat jatkuvasti, joten optimaalisempi ratkaisu tulee olemaan ajan myötä tarpeen. Kasvun odotetaan olevan erityisen suuri, kun peli julkaistaan vuonna 2016 myös iOS-alustalle.

Tiedostojen jakeluun web-palvelimella vaikuttaa moni tekijä. Näitä tekijöitä ovat muun muassa käytetty web-palvelinsovellus, käytetty yhteysprotokolla, välimuistituksen hyödyntäminen ja CDN-palvelun käyttö. On hyvä selvittää, kuinka suuri vaikutus kultakin tekijältä on odotettavissa, että tiedetään kuinka suuren työ- ja kustannusmäärän arvoisia ne ovat.

2 UNITY-PELIMOOTTORI

Unity on Unity Technologiesin kehittämä pelimoottori ja kehitysympäristö, jota käytetään pelien tuottamiseen useille eri alustoille. Sen ensimmäinen versio oli vain Mac-alustalle suunnattu kehitystyökalu. Se julkaistiin vuonna 2005. [1.] Nykyään Unitylla voi kehittää pelejä kymmenille eri alustoille, jotka yritys luokittelee viiteen kategoriaan: mobiili, pelikonsolit, web, desktop-tietokoneet, virtuaalitodellisuus ja lisätty todellisuus, sekä älytelevisiot. [2.]

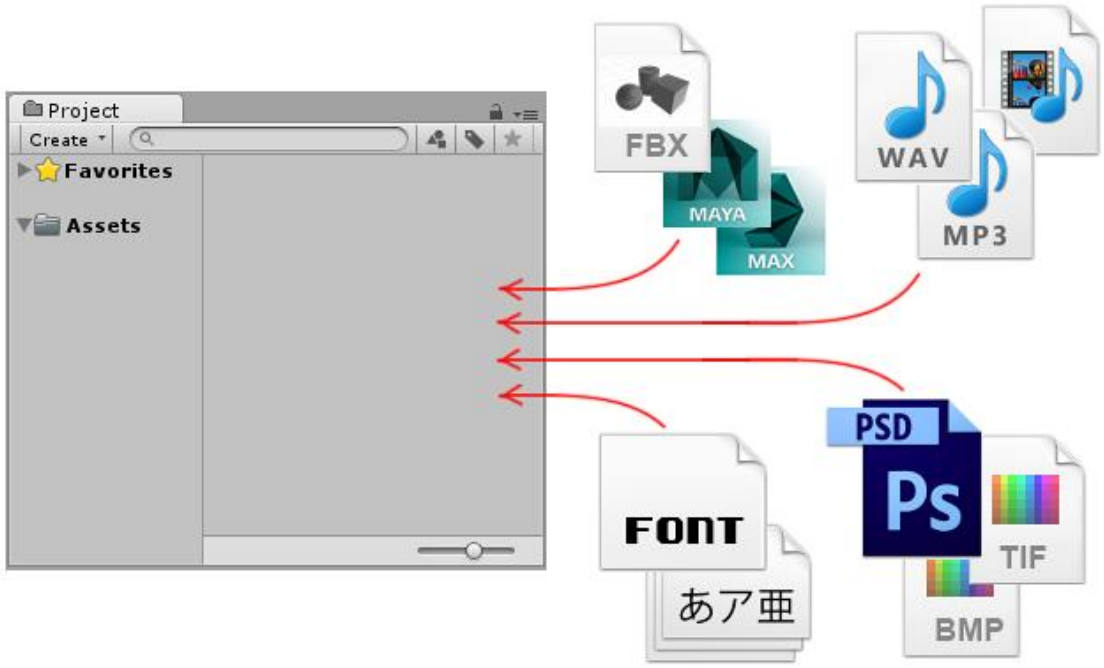
Unityn menestys on osin seurausta vähäresurssisten itsenäisten pelinkehittäjien tukemisesta. Itsenäisillä pelinkehittäjillä ei usein ollut mahdollisuutta kehittää omaa pelimoottoriaan, tai ostaa käyttöoikeutta suurempiin pelimoottoreihin. Unity Technologies keskittyi pelinkehityksen esteettömyyteen, tavoitteena mahdollistaa pelinkehitys mahdollisimman monelle. [1.]

Unityn versio 3 julkaistiin syyskuussa 2010. Kolmas versio keskittyi tuomaan Unity-ympäristöön lisää työkaluja, joita suuremmilla pelistudioilla on usein käytettävissään. Tämän ansiosta pelimoottorin suosio suurempien kehittäjien joukossa kasvoi, ja pienemmät kehitystiimit saivat käyttöönsä entistä kattavamman edullisen pelimoottorin. [1.]

Unity 5 julkaistiin maaliskuussa 2015. Se muun muassa nosti pelimoottorin grafiikan laatua, paransi kehitysympäristön suorituskykyä ja nosti tuettujen alustojen lukumäärän 21:een. Lisäksi pelimoottorista tuotiin saataville Personal Edition, joka kykenee kaikkeen samaan kuin Professional Edition, mutta on tarkoitettu kehittäjille joiden tuoton ja rahoituksen määrä on alle 100 000$. [3.]

2.1 AssetBundlet

Asseteilla tarkoitetaan Unity-pelissä tai -projektissa käytettävissä olevia kohteita, kuten 3D-malleja, äänitiedostoja ja kuvia. Assetit luodaan yleensä Unityn ulkopuolella, mutta joitakin assetteja voi luoda myös Unityssa. [4.]



Kuva 1. Esimerkkejä Unity-pelimoottorin tukemista asseteista [4].

AssetBundlet ovat pakattuja tiedostoja, jotka sisältävät Unity-projektin assetteja. Niitä voi luoda Unity-kehitysympäristössä haluamistaan asseteista. Niiden tarkoitus on yksinkertaistaa Unity-sovellukseen sisällön lataamista verkon yli. Sovellus voi ajon aikana ladata niitä palvelimelta ja käyttää niiden sisältämiä assetteja. [5.]

2.2 AssetBundlejen välimuistitus ja versionhallinta

Unity-sovellus voi ladata AssetBundleja joko hyödyntämättä tai hyödyntäen välimuistitusta. Lataaminen välimuistitusta hyödyntäen tai hyödyntämättä tapahtuu eri komennoilla. AssetBundlet välimuistitetaan Unityn omaan välimuistiin käyttäjän laitteen paikalliseen tallennustilaan. Unity WebPlayer -sovelluksilla on käytettävissään 50 MB kaikkien WebPlayer-sovellusten yhteistä välimuistia. Sovellukselle voi tarvittaessa saada oman suuremman välimuistin, jonka koon määrää Unityn ja kehittäjän välinen lisenssisopimus. Työpöytä- ja mobiilisovelluksilla on käytettävissään 4 GB välimuistitilaa. [6.]

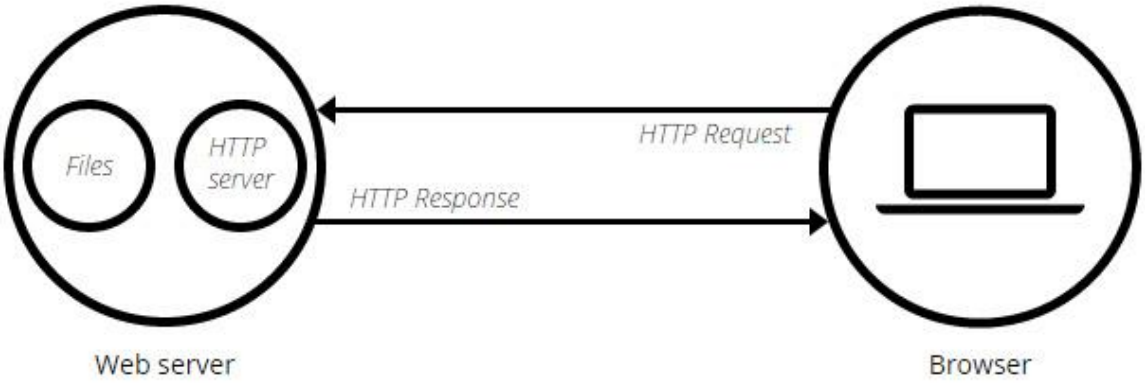
Välimuistitilan loppuessa välimuistista poistetaan AssetBundleja, kunnes siellä on tarpeeksi tilaa uudelle AssetBundlelle. Poistaminen alkaa AssetBundleista, joiden edellisestä käytöstä on kulunut eniten aikaa. Jos tilaa ei ole mahdollista vapauttaa tarpeeksi, AssetBundlea ei välimuistiteta. Näin voi käydä, jos kaikki välimuistissa olevat AssetBundlet ovat käytössä tai paikallinen tallennustila on täynnä. [7.]

AssetBundlejen luonti Unity-editorissa luo kullekin AssetBundlelle oman .manifest-tiedoston. Tästä tiedostosta löytyy muun muassa tiedoston tiivisteen ja tieto sen asset-riippuvuuksista. [8.] Alla esimerkki .manifest-tiedoston sisällöstä.

```
ManifestFileVersion: 0
CRC: 2422268106
Hashes:
 AssetFileHash:
   serializedVersion: 2
   Hash: 8b6db55a2344f068cf8a9be0a662ba15
 TypeTreeHash:
   serializedVersion: 2
   Hash: 37ad974993dbaa77485dd2a0c38f347a
HashAppended: 0
ClassTypes:
- Class: 91
 Script: {instanceID: 0}
Assets:
 Asset_0: Assets/Mecanim/StateMachine.controller
Dependencies: {}
```

3 TIEDOSTOJAKELUN SUORITUSKYKYYN WEB-PALVELIMELLA VAIKUTTAVIA TEKIJÖITÄ

Web-palvelin on sovellus, jonka päätehtävä on tarjoilla verkkosivuja. Se odottaa pyyntöjä asiakasohjelmilta ja lähettää vastauksena pyydetyn datan. Asiakasohjelmana toimiii yleensä verkkoselain. [9.] Visuaalinen esitys tästä on nähtävissä kuvassa 2.



Kuva 2. Visuaalinen esitys web-palvelimen toiminnasta [10].

Web-palvelimella voidaan myös viitata tietokoneeseen, jolla web-palvelinsovellus ja sen tiedostot sijaitsevat. Tällainen tietokone on yleensä jatkuvassa internet-yhteydessä, jotta se on aina saavutettavissa. [9.]

Web-palvelinsovelluksista yleisimmin käytettyjä ovat Apache, Microsoftin Internet Information Server (IIS) ja nginx. Sovelluksen valintaan vaikuttavat muun muassa palvelimen käyttöjärjestelmä, sovelluksen tietoturvallisuus ja sen tuki palvelinpuolen ohjelmoinnille. [11.]

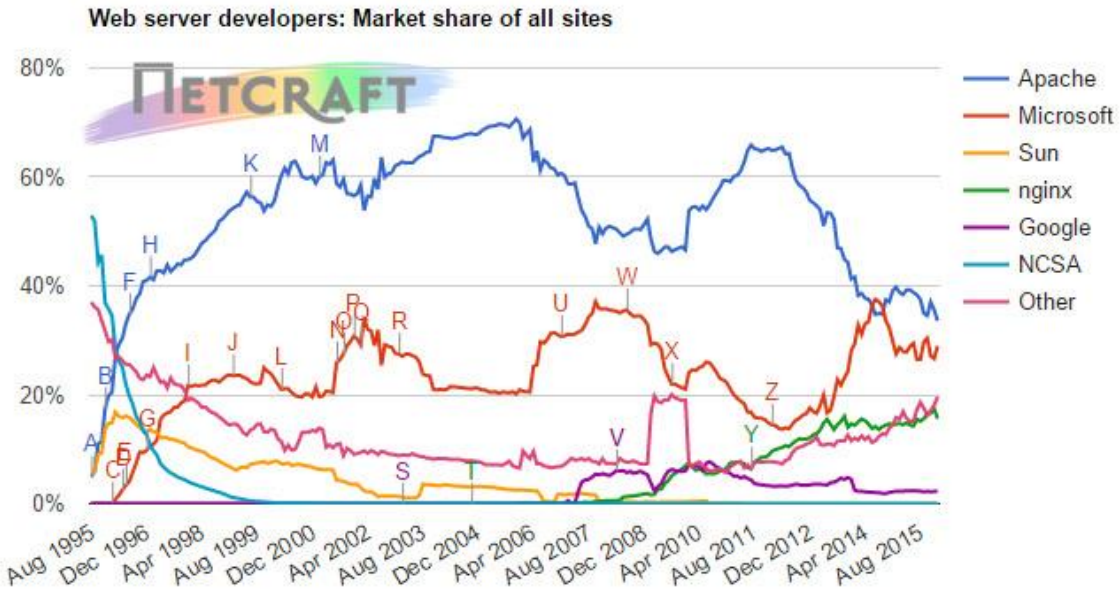
Web-palvelimia voidaan ajatella olevan kahdenlaisia: staattisia ja dynaamisia. Staattinen web-palvelin koostuu vain palvelintietokoneesta ja web-palvelinsovelluksesta. Se tarjoaa asiakasohjelman pyytämät tiedostot alkuperäisessä muodossaan. [10.]

Dynaamisella web-palvelimella on palvelintietokoneen ja web-palvelinsovelluksen lisäksi muitakin sovelluksia. Nämä sovellukset ovat useimmissa tapauksissa tietokanta ja sovelluspalvelin. Dynaamisen web-palvelimen tarjoamat web-sivut

eivät vastaa suoraan palvelimella olevia HTML-dokumentteja. Sen sijaan sovelluspalvelin kokoaa tarjottavan sivun esimerkiksi käyttäen HTML-mallipohjaa ja tietokannan sisältöä. [10.]

3.1 Web-palvelinsovellusvaihtoehdot

Netcraft tekee kuukausittain katsauksen eri web-palvelinsovellusten käyttömäärään. Tammikuun 2016 katsauksessa kolme suosituinta web-palvelinsovellusta olivat Apache, IIS ja Nginx. [12.] Eri web-palvelinsovellusten markkinaosuus noin 900 miljoonasta katsaukseen osallistuneesta verkkosivustosta on nähtävissä kuvassa 3.



Kuva 3. Eri web-palvelinsovellusten markkinaosuus Netcraftin tammikuun 2016 katsauksessa [12].

3.1.1 Apache

Apache HTTP Server on vuoden 1995 huhtikuussa julkaistu web-palvelinsovellus. Sen ensimmäisen virallisen julkisen version pohjana käytettiin silloista suosituinta HTTP daemonia, NCSA httpd 1.3. Apachelle kehitettiin pian oma palvelinarkkitehtuuri, jota käytettiin sovelluksen pohjana vuoden 1995 elokuussa

julkaistusta 0.8.8 versiosta lähtien. Apachen 1.0 versio julkaistiin saman vuoden joulukuussa. [13.]

Apache on ollut suosituin web-palvelinsovellus vuodesta 1996 lähtien. Suosion ansiosta sille on saatavilla paljon dokumentaatiota ja muiden sovellusprojektien tuki sille on laaja. [14.]

Apachen modulaarinen rakenne tekee siitä joustavan. Ylläpitäjät voivat muokata sen tarpeisiinsa sopivaksi käyttäen eri moduuleja. moduuleilla voidaan tuoda sovellukselle natiivi tuki muun muassa URL uudelleenkirjoitukselle ja SSL-salaukselle. Apachessa on valmiiksi mukana monia usein käytettyjä moduuleja. [15.]

3.1.2 Nginx

Nginx on alun perin Igor Sysoevin kehittämä HTTP-palvelin, käänteinen välityspalvelin, sähköpostivälityspalvelin ja yleiskäyttöinen TCP-välityspalvelin [16]. Sen kehitys aloitettiin vuonna 2002, ja ensimmäinen virallinen julkaisu tapahtui vuonna 2004. Se kehitettiin ratkaisuna 10000 yhtäaikaisen yhteyden käsittelylle, mihin mikään muu luotettava ja tuotantokäyttöön soveltuva web-palvelin ei silloin kyennyt. [17.]

Nginx tunnetaan vähäisestä resurssien käytöstään, luotettavuudestaan, suorituskyvystään ja helposta konfiguroinnistaan. Perinteisistä web-palvelimista poiketen se ei käytä yksittäisiä CPU-säikeitä kyselyiden käsittelyyn. Sen sijaan se käyttää paremmin skaalautuvaa tapahtumapohjaista arkkitehtuuria. Se tekee etenkin sovelluksen rasituksen aikaisesta muistin käytöstä pienen ja ennakoitavan. [18.]

3.1.3 Internet Information Services

Internet Information Services on Microsoftin Windows-käyttöjärjestelmiin sisäänrakennettu Web- ja sovelluspalvelin. Sen ensimmäinen versio julkaistiin

Windows NT 3.51:n Service Pack 3 -päivityspaketissa. Uusin 8.0 versio on saatavilla Windows Server 2012 ja Windows 8 -alustoille. [19, s. 3-4.]

IIS:n arkkitehtuuri muuttui modulaariseksi versiossa 7.0. Tämän ansiosta palvelimelle on mahdollista ladata käyttöön vain tarvitut moduulit, jolloin hyökkäyspinta-ala ja muistinkäyttö ovat pienempiä. [19, s. 26.]

3.1.4 Vaihtoehtojen vertailu

Monet ylläpitäjät ovat tehneet testejä verratakseen Apachen ja Nginxin nopeutta ja suorituskykyä. Yleinen linja testien tuloksissa on, että Nginx on Apachea nopeampi. Sen RPS (requests per second) on usein jopa kaksinkertainen Apacheen verrattuna. Vasteajat ovat myös matalampia, ja ero kasvaa Nginxin eduksi suuremmaksi kyselyiden määrän kasvaessa. [20, s. 245.]

Palvelinsovellusten resurssienkäytössä on myös suuri ero. Eräässä esimerkkitapauksessa kymmeniä miljoonia kyselyitä palvellut Nginx käytti vain noin 15 MB muistia ja 10% CPU-kapasiteetista huippurasituksessa. Samanlaisessa rasituksessa Apache alkoi hidastua huomattavasti ja käyttämään paljon enemmän muistia noin 1000 yhtäaikaisen kyselyn kohdalla. [20, s. 246.]

Apachen etuna on joustavuus ja helppokäyttöisyys. Sen pidemmän iän ja suuren suosion ansiosta apua ja dokumentaatiota mahdollisiin ongelmiin on helppoa löytää. Samasta syystä sille on myös saatavilla satoja moduuleja erilaisiin tarpeisiin. Moduuleja on myös helpompi ottaa käyttöön tai poistaa käytöstä, koska siihen tarvitaan vain konfiguraatiomuutos. Nginxillä tällainen dynaaminen moduulien lataus ei ole mahdollista, vaan moduulien pitää olla siinä mukana kääntäessä sen lähdekoodi sovellukseksi. [20, s. 243-244.].

IIS on saatavilla vain Windows-palvelimille, mikä tuo sille lisenssikustannuksia. Mahdollisia syitä sen valinnalle on muun muassa se, että käytössä on jo Windows-palvelin, tai että halutaan hyödyntää sen vahvaa ASP.net integraatiota. [21.] IIS:llä ei näytä olevan vahvuuksia, mitkä olisivat AssetBundle jakelussa eduksi.

Huomattavia eroja web-palvelinsovellusten välillä on huomattavissa vain staattisen sisällön jakelussa. Dynaamisen sisällön tapauksessa erot ovat hyvin pieniä. Esimerkiksi PHP:lla rakennetun Wordpress-sisällönhallintajärjestelmällä toimivan verkkosivuston nopeus on paljon enemmän riippuvainen PHP:sta kuin web-palvelinsovellusvalinnasta, ellei sivustoa välimuistiteta. [22.]

3.2 Välimuistitus

Välimuistituksella tarkoitetaan äskettäin käytetyn tiedon väliaikaista tallentamista myöhempää uudelleen käyttöä varten. Sillä säästetään käyttäjän aikaa ja vähennetään liikennettä verkossa. Yleensä välimuistitus tapahtuu käyttäjän tietämättä siitä. Esimerkiksi käyttäjän palatessa aiemmin vierailemalleen verkkosivulle verkkoselain voi käyttää välimuistissa olevia tiedostoja, sen sijaan että hakisi ne uudestaan palvelimelta. [23.]

Välimuistitusta hyödyntämällä voi vähentää palvelimelle tulevien pyyntöjen määrää. Sen ansiosta palvelin jaksaa hoitaa pyyntöjä useammalta käyttäjältä, ja sovelluksen käyttö on nopeampaa. [24.]

Web-ympäristössä välimuistitus on perinteisesti toiminut pääasiassa hyödyntäen käyttäjän verkkoselaimen välimuistia. Nykyään tärkeämpää on kuinka välimuistitus hoidetaan verkkoliikenteen välietapeilla, CDN-palvelimilla. [24.]

3.2.1 Välimuistitusotsakkeet

Kun palvelin vastaa HTTP-kyselyyn, kyselyn mukana on aina kokoelma HTTP otsaketietoja, jotka kertovan muun muassa vastauksen koon, tyypin ja ohjeita sen välimuistitukseen. Näitä välimuistitusohjeita hyödyntäen voidaan kertoa verkkoselaimelle milloin ja kuinka kauan vastauksen voi välimuistittaa. [25.]

Koska Unity-pelimoottori käyttää AssetBundlejen välimuistitukseen ja versiohallintaan omia ratkaisujaan, HTTP-kyselyjen otsakkeet eivät normaalisti vaikuta välimuistitukseen Unity-pelin tapauksessa. Poikkeuksena on CDN-

palvelun käyttö omien palvelimien ja pelin pelaajien välissä. Silloin HTTP-kyselyn otsakkeet vaikuttavat siihen, kuinka CDN-palvelimet välimuistittavat AssetBundlet.

Tärkeimmät välimuistitusotsakkeet ovat kuvattuna alla.

cache-control: private, max-age=0, no-cache

Tämä otsake määrää kuka voi välimuistittaa vastauksen, millä ehdoin ja kuinka kauan. Private määrite kertoo vastauksen olevan tarkoitettu yhdelle tietylle käyttäjälle. Sen voi silloin välimuistittaa käyttäjän oma laite, mutta ei CDN-palvelin. Max-age määrite kertoo sekunteina kuinka kauan vastauksen saa välimuistittaa ja käyttää uudelleen. No-cache määrite kertoo, että tiedostoa voi käyttää uudestaan vasta, kun palvelimelta on kysytty onko se muuttunut viime kerrasta. No-cache määritteen voi korvata myös no-store määritteellä, joka kieltää tiedoston välimuistittamisen kokonaan. [25.]

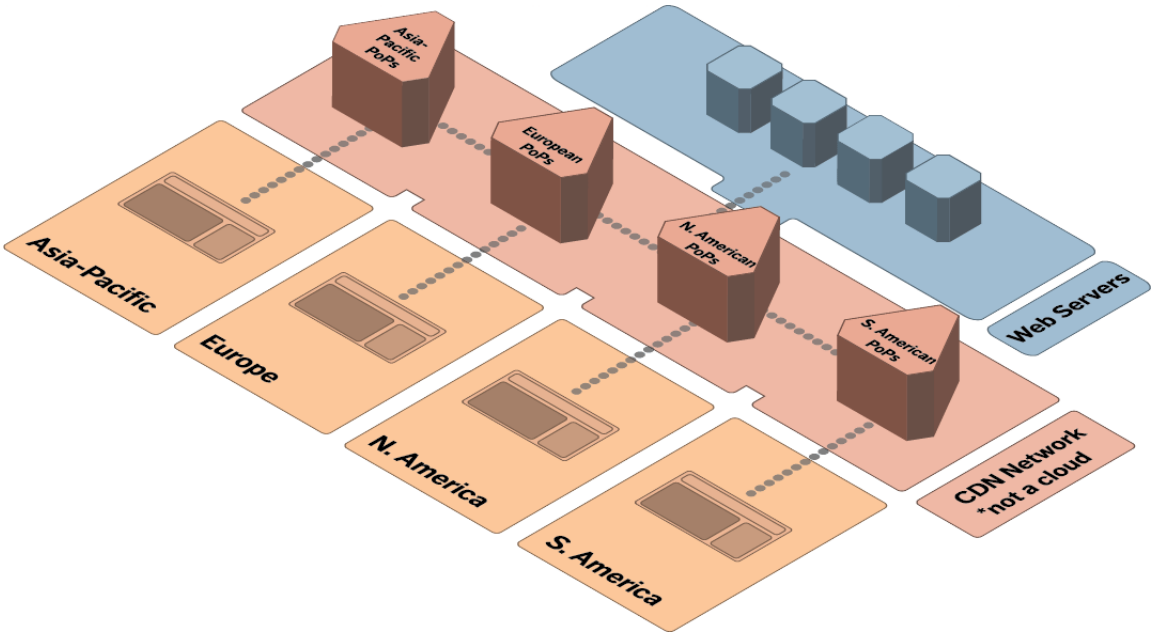
ETag: “x234dff”

Etag-otsake sisältää palvelimen generoiman arvon, esimerkiksi tiedoston sisällöstä generoidun tiivisteen. Jos otsakkeen arvo ei ole muuttunut edellisestä pyynnöstä samaan tiedostoon, tiedosto ei ole muuttunut ja uutta versiota ei tarvitse ladata. [25.]

3.2.2 CDN-palvelun käyttö

CDN-palvelimet ovat palvelimia, jotka ovat omien palvelimien ja käyttäjien yhteyden välissä. Niiden tehtävä on välimuistittaa sisältöä noudattaen HTTP otsakkeiden määrittämiä välimuistitussääntöjä. [24.]

CDN-palvelimet pyrkivät toimittamaan tiedostot käyttäjille palvelimelta, joka on heitä lähimpänä ja johon heillä on nopein yhteys [24]. Tämä on visualisoituna kuvassa 4.



Kuva 4. Visuaalinen esitys CDN-palvelimien roolista [24].

3.3 Yhteysprotokolla

Tiedon välitys internetin yli tapahtuu pääasiassa käyttäen HTTP-protokollaa. HTTP on lyhenne sanoista Hypertext Transfer Protocol. Se on tapa muotoilla tietoa ja välittää sitä asiakasohjelman ja web-palvelimen välillä. [26.]

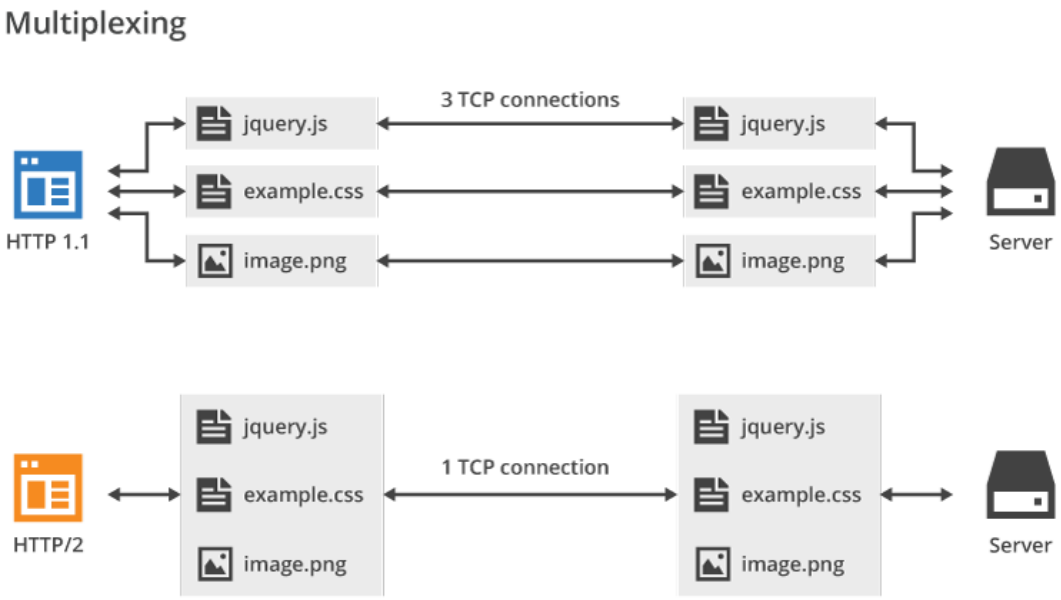
HTTP toimii käyttäen kysely-vastaus kaavaa, jossa palvelimen vastauksessa asiakasohjelman kyselyyn on pyydetyn sisällön lisäksi olennaista tietoa kyselystä. Tämän ansiosta yhteyden välietappeina toimivat laitteet ja palvelut osaavat käsitellä kyselyitä ja voivat soveltaa niihin esimerkiksi kuormantasausta, välimuistitusta, salausta ja pakkausta. [26.]

3.3.1 HTTP/2 ja SPDY

HTTP/2 hyväksyttiin uudeksi standardiksi helmikuussa 2015. Sitä edeltävä HTTP/1.1 oli ollut käytössä yli 15 vuotta. Uuden standardin on tarkoitus parantaa suorituskykyä ja vähentää etenkin käyttäjän kokemaa viivettä käyttäessään

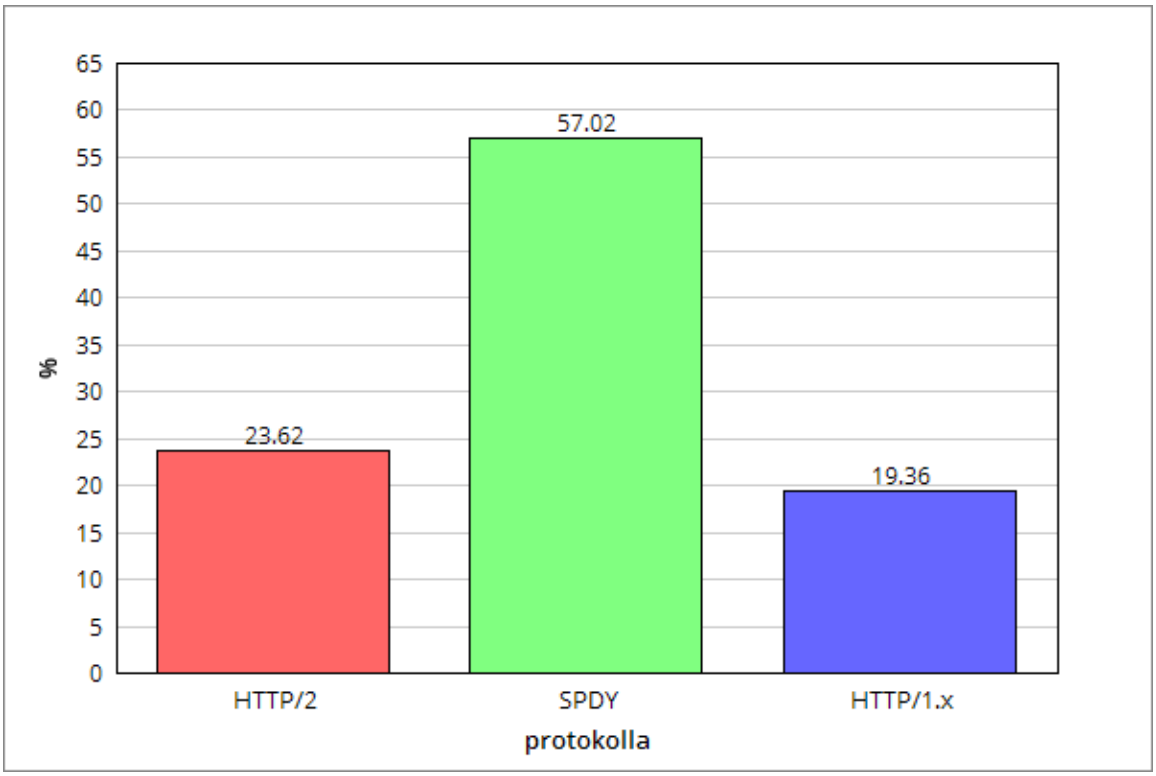
verkkoselainta. Toissijainen tarkoitus on tehostaa verkon ja palvelinten resurssien käyttöä. [27.]

Yksi HTTP/2-protokollan suuri hyöty on, että usean kohteen lataamiseen riittää yksi TCP-yhteys. Tämä on esitettynä kuvassa 5. Vastausten ei myöskään tarvitse enää saapua pyyntöjärjestyksessä. Lisäksi asiakasohjelma voi kertoa kyselyissään palvelimelle mitkä haluamansa resurssit se näkee muita tärkeämmiksi, ja HTTP otsakkeet pakataan, pienentäen kyselyiden ja vastausten kokoa. [27.]



Kuva 5. Esimerkki HTTP/1.1- ja HTTP/2-protokollien tarvitsemista TCP yhteyksistä verkkosivun lataamiseen [27].

Useiden laajasti käytettyjen verkkoselainten uusimmat versiot tukevat HTTP/2-protokollaa. Joulukuun 2015 kahtena ensimmäisenä päivänä Cloudflaren verkkosivulle kohdistuneesta liikenteestä 26,79% tuli HTTP/2-tuella varustetuista selaimista. Kun Cloudflaren verkkosivun HTTP/2-tuki oli ollut käytössä yli kaksi vuorokautta, vierailuista yli puolet tapahtui edelleen käyttäen SPDY-protokollaa. Eri protokollien osuus vierailuista on tarkemmin esitettynä kuvassa 6. Näiden tilastojen pohjalta Cloudflare ei nähnyt SPDY-tuen poistamista vielä järkeväksi. [27.]



Kuva 6. HTTP/2, SPDY ja HTTP/1.x –protokollien osuus Cloudflaren verkkosivun vierailuista HTTP/2-tuen kahtena ensimmäisenä päivänä.

HTTP/2 perustuu suurelta osin Googlen kehittämään SPDY protokollaan, jonka ensimmäinen julkistus oli marraskuussa 2009 [26]. Google aikoo poistaa Chrome-selaimestaan SPDY-tuen vuonna 2016, antaen HTTP/2-protokollan korvata sen [28]. Myös Nginx antaa SPDY-protokollan väistyä HTTP/2-protokollan tieltä: Nginxin SPDY moduuli ei ole käytettävissä samanaikaisesti HTTP/2 moduulin kanssa [27].

HTTP/2 toi SPDY-protokollaan nähden useita parannuksia. Yhden yhteyden riittämistä usean kohteen lataamiseen voi hyödyntää usean palvelimen kanssa samanaikaisesti. Salatut yhteydet toimivat nopeammin. Otsakkeiden pakkaus on tehokkaampaa ja turvallisempaa HTTP otsakkeiden pakkaukseen tarkoitetun HPACK-pakkausmuodon ansiosta. Lisäksi protokolla ei enää vaadi salauksen käyttöä. Käytännössä se on kuitenkin usein vaadittua, koska monet suositut verkkoselaimet vaativat salatun yhteyden käyttääkseen HTTP/2-protokollaa. [28.]

3.3.2 Protokollien nopeusvertailu

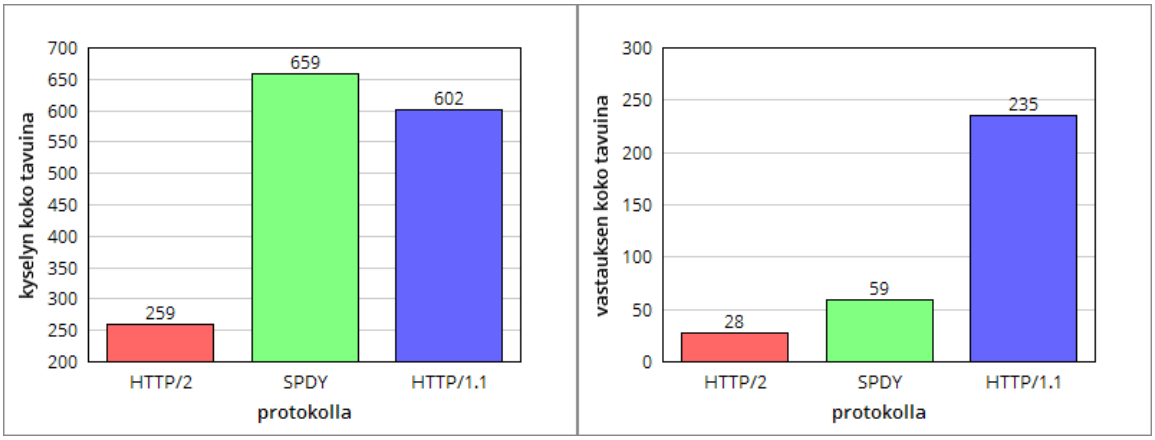
HttpWatch vertasi HTTP/1.1-protokollan, HTTP/2-protokollan ja SPDY/3.1-protokollan nopeutta Iso-Britannian Google-sivustolla. Vertailu suoritettiin poistamalla Firefox-selaimen asetuksista tuki kaikille muille protokollille paitsi kulloinkin testattavalle. Kukin mittaus tehtiin uudella Firefox-istunnolla ja tyhjällä välimuistilla. Vertailussa käytetyssä selaimessa oli käytössä HTTP/2-protokollan luonnosversio, joka ei kuitenkaan merkittävästi eroa lopullisesta versiosta. [29.]

Vertailun lopputuloksena todettiin HTTP/2-protokollan tuovan huomattavia suorituskykyparannuksia HTTP/1.1-protokollaan ja jopa SPDY-protokollaan nähden [29]. Tarkemmat tulokset neljästä vertailun osa-alueesta ovat alla.

Kysely- ja vastausotsakkeiden koko

Toisin kuin HTTP/1.1, SPDY ja HTTP/2 pakkaavat HTTP otsakkeet. SPDY käyttää pakkaukseen DEFLATE algoritmia, joka on suunniteltu moneen käyttöön soveltuvaksi. HTTP/2 käyttää HPACK algoritmia, joka on suunniteltu juurikin HTTP otsakkeiden pakkaukseen. [29.]

Vertailuun käytettiin Googlen sivustolla ajoittain tapahtuvaa sisällötöntä kyselyä. Vertailun tulokset ovat nähtävissä kuvassa 7. HTTP/2-protokolla osoittautui selvästi nopeimmaksi 259B kokoisella kyselyllä ja 28B kokoisella vastauksella. [29.]



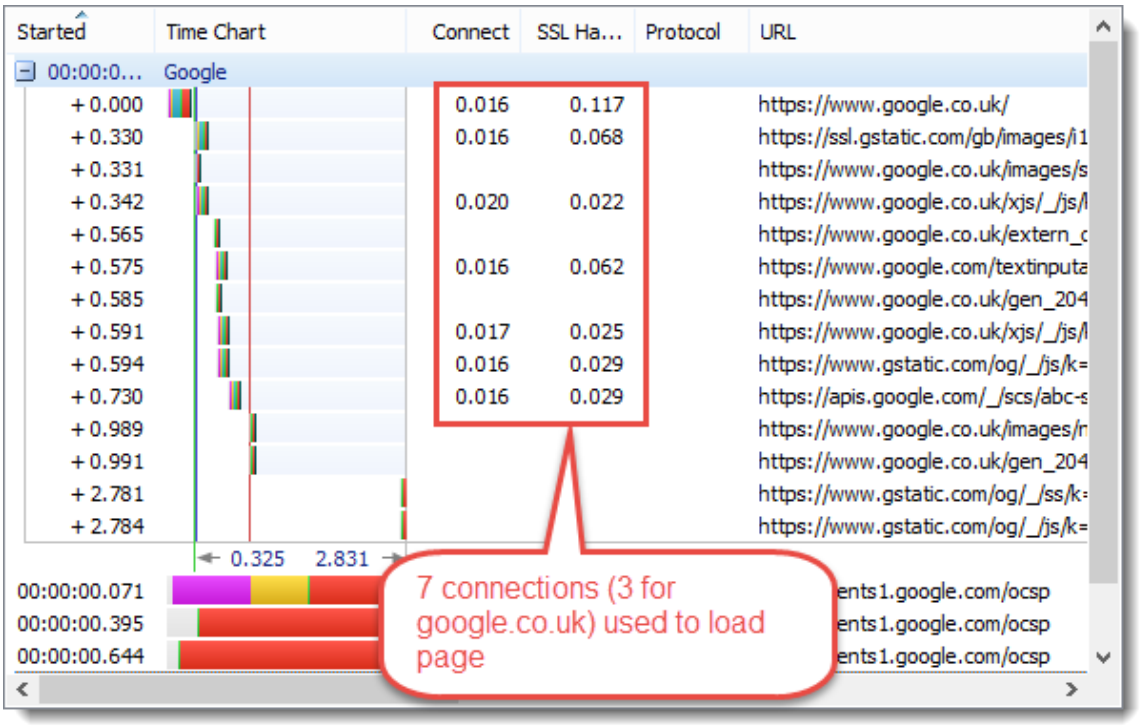
Kuva 7. www.google.co.uk-sivulla ajoittain tapahtuvan sisällöttömän kyselyn ja sen vastauksen koko tavuina eri protokollilla.

Vastauksen koko

Koska HTTP/2-protokollan otsakkeet ovat pienimmät, kuvatiedoston palauttavan kyselyn vastaus oli odotetusti pienin käyttäen HTTP/2-protokollaa. Tekstipohjaisen resurssin kohdalla SPDY-protokollan vastaus osoittautui pienimmäksi. Syy oli siinä, että Googlen palvelin lisää HTTP/2-protokollalla kuljetettavaan tekstidataan täytetavuja. Täytetavut ovat HTTP/2-protokollan valinnainen toiminto, jolla voi hämärtää datan täsmällistä kokoa ja siten estää tietynlaisia hyökkäyksiä. [29.]

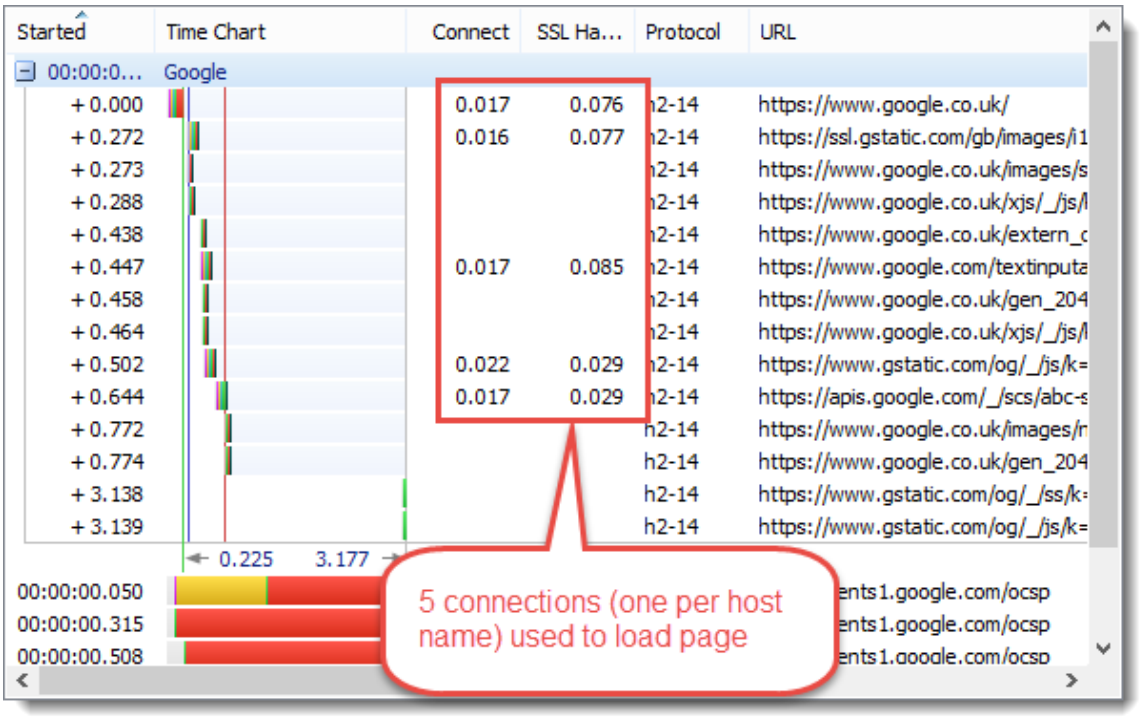
TCP-yhteyksien ja SSL-kättelyiden määrä sivun latauksessa

HTTP/1.1 paransi latauksien samanaikaisuutta ja verkkoyhteyden käytön tehokkuutta sallimalla kuusi tai enemmän samanaikaista yhteyttä samaan osoitteeseen. Kullakin erillisellä yhteydellä on kuitenkin omat suorituskykykustannuksensa ja SSL-kättelyt. [29.] Esimerkki useasta yhteydestä samaan osoitteeseen HTTP/1.1-protokollalla on esitettynä kuvassa 8.



Kuva 8. www.google.co.uk-sivun lataukseen tarvittujen yhteyksien määrä HTTP/1.1-protokollalla [29].

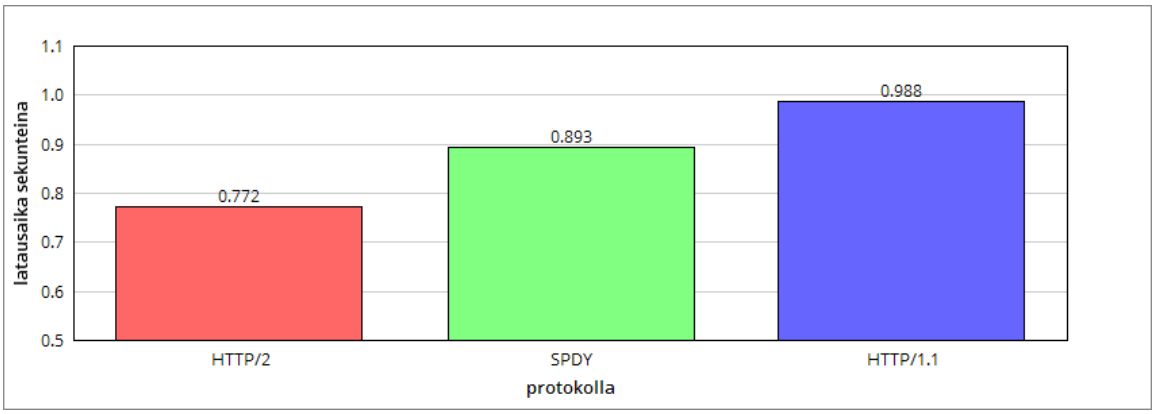
HTTP/2- ja SPDY-protokollat tukevat samanaikaisuutta yhdellä yhteydellä kuhunkin osoitteeseen käyttäen multipleksointia. Multipleksoinnin ansiosta yksi yhteys palvelimeen riittää usean kyselyn verkkoliikenteen kuljettamiseen. [29.] Iso-Britannian Google-sivuston lataukseen HTTP/2-protokollalla tarvittujen yhteyksien määrä on esitettynä kuvassa 9.



Kuva 9. www.google.co.uk-sivun lataukseen tarvittujen yhteyksien määrä HTTP/2-protokollalla [29].

Sivun latausaika

Vertailun testiympäristö ilmoittaa ajan milloin sivu oli kokonaisuudessaan ladattu ja valmis käytettäväksi. Vertailun tulokset ovat nähtävissä kuvassa 10. Nopein 0,772 sekunnin tulos oli HTTP/2-protokollalla. [29.]



Kuva 10. www.google.co.uk-sivun latausaika eri protokollilla.

HttpWatch arvioi otsakkeiden pakkauksen ja pienemmän tarvittujen TCP-yhteyksien ja SSL-kättelyjen määrän olevan suurin syy nopeuseroon SPDY- ja HTTP/2-protokollien eduksi. Lisäksi he uskovat eron olevan vielä suurempi monimutkaisempien sivustojen tapauksessa. [29.]

3.4 HTTP pakkaus

HTTP pakkauksen avulla sisältöä voidaan pakata palvelimella ennen sen lähettämistä asiakasohjelmalle. Pakkauksella voidaan pienentää vastauksen kokoa huomattavasti, vähentäen verkon käyttöä ja nopeuttaen latausaikoja. Hyöty on erityisen suuri tekstimuotoisen sisällön kohdalla. Jo pakattujen tiedostojen, kuten gif-kuvien ja zip-tiedostojen, tapauksessa pakkaamisesta ei yleensä ole hyötyä. Lisäpakkauksen yrittäminen voi kasvattaa tiedoston kokoa ja haaskata palvelimen suoritinaikaa. [30.]

HTTP pakkaukseen käytetään yleisimmin joko deflate tai gzip algoritmia. Palvelin ja asiakasohjelma sopivat käytettävän algoritmin käyttäen HTTP otsakkeita. Accept-Encoding otsake kyselyssä kertoo palvelimelle mitä pakkausalgoritmeja asiakasohjelma tukee. Content-Encoding otsake vastauksessa kertoo asiakasohjelmalle mitä algoritmia vastauksen pakkaukseen on käytetty. Asiakasohjelma tietää siten mitä algoritmia sen täytyy käyttää vastauksen purkamiseen. [30.]

4 ASSETBUNDLE JAKELUN OPTIMOINTI

Edellisessä kappaleessa kuvatut tiedostojakelun suorituskykyyn vaikuttavat tekijät kuvailtiin pääasiassa verkkosivustojen nopeuden optimoinnin näkökulmasta. Kaikki niistä eivät ole sovellettavissa AssetBundle jakeluun. Tässä kappaleessa käydään läpi Critical Opsin nykyinen AssetBundle jakeluratkaisu ja sen edut sekä heikkoudet. Tämän jälkeen käydään läpi kuinka edellisessä kappaleessa kuvatut tekijät olisivat sovellettavissa AssetBundle jakelun optimointiin.

4.1 Critical Opsin nykyinen AssetBundle jakeluratkaisu

Critical Opsin AssetBundle jakelun hoitavat nykyisellään samat palvelimet jotka pyörittävät myös pelin jatkuvaa verkkoliikennettä ja sen palvelinpuolen logiikkaa. Palvelimia on viidellä eri alueella, joilla on pelin suurimmat pelaajakeskittymät. Palvelimilla käytetty web-palvelinsovellus on IIS.

Nykyinen ratkaisu ei ole optimaalinen useasta syystä. Pelin jatkuvan verkkoliikenteen kuljetus ja käsittely vaatii hyvin erilaisia vahvuuksia palvelimelta kuin AssetBundle jakelu. Pelin jatkuvan verkkoliikenteen käsittely tarvitsee mahdollisimman pientä verkon viivettä ja paljon suoritintehoa. AssetBundle jakelussa pienen viiveen sijaan tärkeää on suuri kaistanleveys. Tästä syystä nämä tehtävät olisi hyvä erottaa eri palvelimille.

Web-palvelinsovelluksena käytetty IIS ei ole staattisten tiedostojen jakeluun sopivin. Nginx kykenisi palvelemaan suurempaa määrää kyselyjä vähemmällä resurssienkäytöllä. Tällä ei kuitenkaan ole pelin nykyisellä pelaajamäärä vielä suurta merkitystä.

AssetBundlejen pitäminen ajan tasalla ja synkronoituna usealla palvelimella luo hieman lisätyötä nykyisellä jakeluratkaisulla. CDN-palvelua käyttäessä AssetBundlet tarvitsisi pitää ajan tasalla vain yhdellä palvelimella. CDN-palvelu noutaisi tiedostot omille palvelimilleen aina niiden muuttuessa ja hoitaisi

alueellisen jakelun luotettavasti ja nopeasti. CDN-palvelu olisi kuitenkin nykyiseen ratkaisuun verrattuna huomattavasti kalliimpi.

4.2 Optimaalinen AssetBundle jakelu aiempaan selvitykseen pohjautuen

Tässä osiossa käydään läpi miten aiemmin kuvatut tiedostojakelun suorituskykyyn web-palvelimella vaikuttavat tekijät ovat sovellettavissa AssetBundle jakeluun. Lisäksi arvioidaan joidenkin niistä taloudellista kannattavuutta nykyiseen ratkaisuun nähden.

4.2.1 Web-palvelinsovelluksen valinta

Unityn AssetBundlet ovat web-palvelimen näkökulmasta staattista sisältöä. Tästä syystä web-palvelinsovellukseksi kannattaa valita Nginx, joka on yleisesti todettu siihen tarkoitukseen sopivimmaksi vaihtoehdoksi. Staattisten tiedostojen jakelu on web-palvelimen yksinkertaisimpia käyttötarkoituksia, joten Apachen modulaarisuuden tuomalla joustavuudella ja sen suuremmalla dokumentaation määrällä ei ole merkittävää hyötyä tässä käyttötarkoituksessa.

Web-palvelinsovellusvertailut ovat yleensä verkkosivukäytön näkökulmasta tehtyjä. Verkkosivun latauksessa palvelimelta ladattavat tiedostot ovat yleensä lukuisia, mutta pieniä. Unityn AssetBundlet ovat niihin nähden hieman suurempia, joten toistuvissa operaatioissa, kuten yhteyden aloituksessa, saavutetun lisänopeuden vaikutus ei ole niin merkittävä. Tästä syystä käytännön eron palvelinsovellusten suorituskyvyn välillä AssetBundle jakelussa ei pitäisi olla niin suuri kuin testit antavat ymmärtää.

4.2.2 Välimuistitus

Välimuistituksella odotettiin olevan suuri vaikutus AssetBundle-jakelun optimoinnissa. AssetBundlet ovat pääasiassa melko suuria tiedostoja, jotka

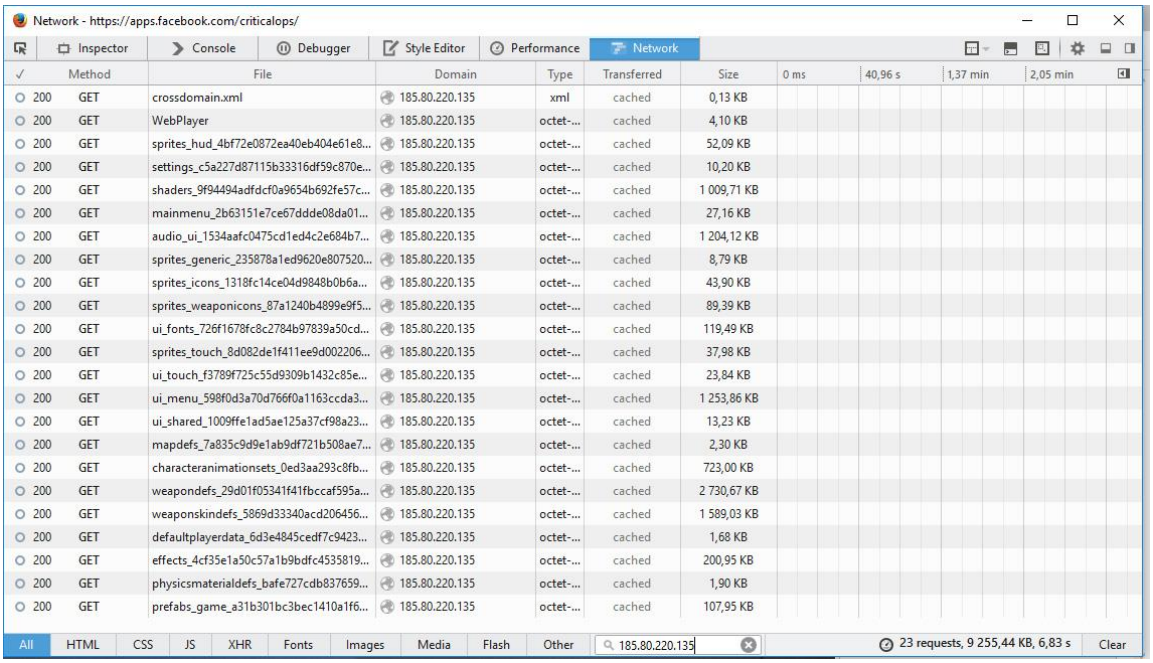
yleensä eivät muutu kovin usein. Niitä olisi siksi hyvä voida välimuistittaa käyttäjän laitteella palvelimen kapasiteetin ja verkkokaistan säästämiseksi. Palvelin jaksaisi siten palvella useampaa käyttäjää ja verkkokaistaa ei tarvittaisi niin paljon. Molemmat hyödyt osaltaan vähentävät palvelinkustannuksia.

Koska Unity käyttää AssetBundlejen versiohallintaan ja välimuistitukseen omia ratkaisujaan, välimuistituksen toimintaa ei voi hallita välimuistitusotsakkeilla. Tätä mahdollisuutta ei ole kuitenkaan kaivattu, koska Unityn oma ratkaisu on toiminut luotettavasti. Se ei myöskään vaadi kehittäjiltä lisätyötä välimuistituksen eteen, koska tiivisteet ja muut siihen tarvittavat tiedot generoidaan automaattisesti luodessa AssetBundle Unity-kehitysympäristössä.

Välimuistituksen vaikutuksen testaus

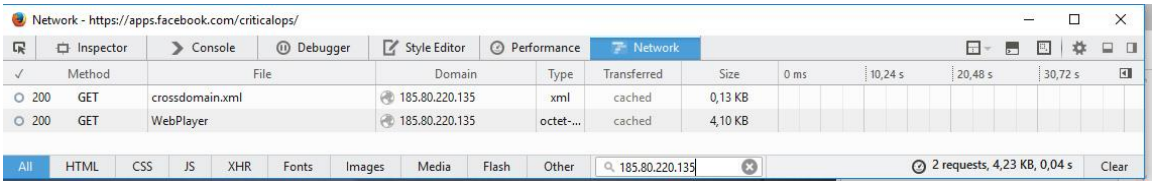
Välimuistituksen vaikutusta testattiin pelin Facebook-versiossa. Testissä seurattiin Firefox-verkkoselaimen kehitystyökaluja käyttäen pelin verkkoliikennettä. Unityn välimuisti tyhjennettiin ennen testin suorittamista. Pelin annettiin latautua sen päävalikkoon asti kahdesti, jolloin jälkimmäisellä latauskerralla siihen pisteeseen mennessä tarvittavat AssetBundlet ovat Unityn välimuistissa.

Pelin ensimmäisellä käynnistyskerralla AssetBundle-palvelimelle tehtiin 23 kyselyä. Tapahtunut verkkoliikenteen määrä oli kokonaisuudessaan 9255,44 KB, ja sen lataamiseen kului 6,83 sekuntia. Kuvankaappaus testistä on nähtävissä kuvassa 11.



Kuva 11. Pelin pääsyyn mennessä tapahtuneet AssetBundle-kyselyt.

Toisella käynnistyskerralla Peli selvästi hyödynsi välimuistitusta. Kyselyitä AssetBundle-palvelimelle tapahtui vain 2. Tapahtuneen verkkoliikenteen määrä oli 4,23 KB ja lataukseen meni 0,04 sekuntia. Välimuistituksella säästettiin palvelimen kapasiteettia ja verkkoliikennettä sekä pelaajan aikaa. Kuvankaappaus testistä on nähtävissä kuvassa 12.



Kuva 12. Pelin päävalikkoon pääsyyn mennessä tapahtuneet AssetBundle-kyselyt toisella käynnistyskerralla.

4.2.3 Yhteysprotokolla

HTTP/2- ja SPDY-protokollien parannukset HTTP/1.1-protokollaan ovat pääasiassa suunnattuja verkkosivukäyttöön. Tästä syystä monet niistä eivät ole eduksi AssetBundle jakelussa. Suurin hyöty olisi otsakkeiden pakkauksella ja yhden TCP-yhteyden riittämisellä usean kohteen lataamiseen samasta osoitteesta.

Unity-pelimoottori vaikuttaa tukevan kolmesta tässä työssä kuvatusta protokollasta vain HTTP/1.1-protokollaa. Unityn dokumentaatiosta ja päivitysmerkinnöistä ei löytynyt mitään mainintaa tuettujen protokollien muutoksesta viime vuosina. Tämän perusteella voi olettaa, että Unity käyttää verkkoliikenteeseen pitkään standardina ollutta HTTP/1.1-protokollaa. Tämän voi todeta myös kuvassa 13 esimerkkinä olevan AssetBundle-kyselyn perustiedoista.

Kuva 13. Firefoxin kehittäjätyökaluista kuvankaapattu esimerkki AssetBundle-kyselyn perustiedoista, joista näkee siihen käytetyn HTTP/1.1-protokollaa.

4.2.4 HTTP pakkaus

Unityn AssetBundlet ovat jo valmiiksi pakattuja luodessa ne oletusasetuksilla. HTTP pakkaus ei siksi ole sovellettavissa AssetBundle jakelussa. Sen yrittäminen haaskaisi vain palvelimen suoritinaikaa ja ei vähentäisi tapahtuvaa verkkoliikennettä lainkaan.

4.2.5 CDN-palvelun käyttö

CDN-palvelujen pääasiallinen tarkoitus on staattisen sisällön jakelu, joten AssetBundle jakelussa niiden hyödyntäminen on eduksi. CDN-palveluilla on palvelimia ympäri maailmaa, joten palvelun hyödyntäminen takaisi vikasietoisen ja nopean AssetBundle jakelun mahdollisimman suurelle osalle pelin pelaajista.

Edullisempi ja lähes yhtä hyvä ratkaisu on ylläpitää omia AssetBundle jakelupalvelimia alueilla, joilla on pelin pelaajamäärän suurimmat keskittymät. Hyvin tehtynä tällä ratkaisulla keskimääräisten pelaajien kokemien latausaikojen ei pitäisi olla juurikaan hitaampia kuin CDN-palvelua käyttäessä. Tämä ratkaisu vaatii kuitenkin paljon enemmän ylläpitotyötä. Lisäksi se ei ole yhtä vikasietoinen

kuin CDN-palvelu jos jakelupalvelimia on vain yksi kullakin alueella. Pelaajamäärien kasvaessa hyvin suuriksi vikasietoisen ja riittävän nopean palvelun takaaminen voi tulla todella kalliiksi tälläkin ratkaisulla.

5 POHDINTA

Tämä opinnäytetyö saavutti hyvin tavoitteensa. Erilaisten seikkojen osuus AssetBundle jakelun optimoinnissa selkiytyi. Lisäksi niihin panostamisen kannattavuuden arviointi on tämän tiedon pohjalta helpompaa. Työn lopputuloksena ei juurikaan jäänyt kysymyksiä siitä, mikä on optimaalinen tapa toteuttaa AssetBundle jakelu. Jäljelle jääneet kysymykset ovat enimmäkseen eri tekijöiden optimoinnin kannattavuudesta niiden kustannuksiin nähden.

Työn tulokset olivat enimmäkseen odotettuja. Esimerkiksi välimuistituksen vaikutus ei juurikaan yllättänyt. Koska AssetBundlet eivät päivity kovin usein, välimuistitus on niihin hyvin sovellettavissa. Sekin oli odotettua, että useimmat tutkituista AssetBundle jakelun tehokkuuteen vaikuttavista tekijöistä ovat oikeasti tärkeitä vasta hyvin suurten pelaajamäärien kanssa. Poikkeuksena on jakelun vikasietoisuus, joka olisi hyvä saada paremmaksi ennen kuin mahdollisen pitkittyneen vikatilanteen kustannukset kasvavat.

Työn tutkimusvaiheessa Unityn oma välimuistitusratkaisu hieman yllätti. Aluksi tuntui, että se olisi epäluotettava verrattuna välimuistituksen hallintaan välimuistitusotsakkeilla, kuten web-ympäristössä tyypillisesti tehdään. Lopulta se vaikutti kuitenkin enimmäkseen positiiviselta. Se toimii luotettavasti ja sen ansiosta välimuistituksen eteen ei tarvitse juurikaan tehdä erillistä työtä. Lisäksi ratkaisu toimii täsmälleen samalla tavalla joka alustalla, mikä on aina ideaalinen tilanne monialustaisuudessa.

Välimuistituksessa voi tulla tulevaisuudessa huolenaiheeksi se, kuinka paljon pelaajien laitteiden paikallista tallennustilaa on järkevää hyödyntää. WebPlayer-ympäristössä tämä ei ole ongelma, koska siellä välimuistin koko on rajattu 50 megatavuun, mikä on hyvin pieni määrä tietokoneiden kovalevyillä nykyaikana. Ongelmaksi voi muodostua tallennustila mobiililaitteilla, joilla Unityn välimuistin rajoitus on hyvin korkea 4 GB. Määrä vastaa monilla mobiililaitteilla yli puolta käytettävistä olevasta tallennustilasta, joten sitä ei kannata kokonaan hyödyntää. Muun muassa tästä syystä pelin assettien ja siten AssetBundlejen koko otetaan pelin kehityksessä huomioon.

Yksi asia, mikä olisi parantanut tämän työn tulosten arvoa, olisi tarkempi tutkimus ja testaus tarvittavasta palvelinkapasiteetista eri pelaajamäärillä. Pelaajamäärän kasvuun voisi tämän tiedon avulla reagoida ennakoivasti, eikä reaktiivisesti nykyisen palvelinkapasiteetin jo loppuessa. CDN-palvelun käytön kannattavuutta voisi myös arvioida paremmin. Näiden asioiden kunnollinen testaus olisi kuitenkin vaatinut muutoksia pelin nykyiseen AssetBundle jakeluratkaisuun, jotta palvelinkapasiteetin tarve pelin todellisilla pelaajamäärillä olisi paremmin verrattavissa suunniteltuun optimaaliseen ratkaisuun. Muutosten tekeminen ei ollut tässä vaiheessa mahdollista.

LÄHTEET

1 History of the Unity Engine [Freerunner 3D Animation Project] saatavilla: https://seraphinacorazza.wordpress.com/2013/02/14/history-of-the-unity-engine-freerunner-3d-animation-project/. Haettu 13/4/2016. 2013.

2 Unity – Multiplatform – Publish your game to over 10 platforms saatavilla: https://unity3d.com/unity/multiplatform. Haettu 13/4/2016.

3 Unity 5 Launch – Unity Blog saatavilla: http://blogs.unity3d.com/2015/03/03/unity-5-launch/. Haettu 13/4/2016. 2015.

4 Unity – Manual: Asset Workflow saatavilla: http://docs.unity3d.com/Manual/AssetWorkflow.html. Haettu 13/4/2016. 2016.

5 Unity – Manual: AssetBundles saatavilla: http://docs.unity3d.com/Manual/AssetBundlesIntro.html. Haettu 13/4/2016. 2016.

6 Unity – Manual: Downloading AssetBundles saatavilla: http://docs.unity3d.com/Manual/DownloadingAssetBundles.html. Haettu 13/4/2016. 2016.

7 Unity – Scripting API: WWW.LoadFromCacheOrDownload saatavilla: http://docs.unity3d.com/ScriptReference/WWW.LoadFromCacheOrDownload.html. Haettu 13/4/2016. 2016.

8 Unity – Manual: Building AssetBundles saatavilla: http://docs.unity3d.com/Manual/BuildingAssetBundles.html. Haettu 13/4/2016. 2016.

9 What is a web server? saatavilla: http://www.webdevelopersnotes.com/basics/what_is_web_server.php. Haettu 8/12/2015.

10 What is a web server? saatavilla: https://developer.mozilla.org/en-US/Learn/Common_questions/What_is_a_web_server. Haettu 8/12/2015.

11 Lingan James. What is web server? saatavilla: http://whatis.techtarget.com/definition/Web-server. Haettu 8/12/2015.

12 January 2016 Web Server Survey saatavilla: http://news.netcraft.com/archives/2016/01/26/january-2016-web-server-survey.html. Haettu 11/2/2016. 2016.

13 About the Apache HTTP Server Project saatavilla: https://httpd.apache.org/ABOUT_APACHE.html. Haettu 17/2/2016

14 Apache vs Nginx: Practical Considerations saatavilla: https://www.digitalocean.com/community/tutorials/apache-vs-nginx-practical-considerations. Haettu 17/2/2016. 2015.

15 Apache Modules Explained saatavilla: http://kb.liquidweb.com/apache-modules-explained/. Haettu 17/2/2016.

16 Nginx saatavilla: http://nginx.org/en/. Haettu 17/2/2016.

17 Mobily Tony. Interview with Igor Sysoev, author of Apache's competitor NGINX saatavilla: http://www.freesoftwaremagazine.com/articles/interview_igor_sysoev_author_apaches_competitor_nginx. Haettu 17/2/2016. 2012.

18 NGINX Wiki saatavilla: https://www.nginx.com/resources/wiki/. Haettu 17/2/2016.

19 Schaefer Kenneth, Cochran Jeff, Forsyth Scott. Professional Microsoft IIS 8. 2012.

20 Nedelcu Clement. Nginx HTTP Server: Adopt Nginx for Your Web Applications to Make the Most of your Infrastructure and Serve Pages Faster than Ever. 2010.

21 Seymour Gail. Which Web Server: IIS vs. Apache saatavilla: http://www.hostway.com/blog/which-web-server-iis-vs-apache/. Haettu 17/2/2016. 2013.

22 Mombrea Matthew. Nginx vs. Apache: Choosing a Linux web server saatavilla: http://www.itworld.com/article/2695421/choosing-a-linux-web-server-nginx-vs-apache.html. Haettu 17/2/2016. 2014.

23 What is caching? saatavilla: https://www.citrix.com/glossary/caching.html. Haettu 15/4/2016.

24 Young Kyle. A Beginner's Guide to HTTP Cache Headers saatavilla: http://www.mobify.com/blog/beginners-guide-to-http-cache-headers/. Haettu 15/4/2016. 2013.

25 Grigorik Ilya. HTTP Caching | Web Fundamentals saatavilla: https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/http-caching. Haettu 15/4/2016.

26 What is HTTP? Protocol for Internet Data Transfer saatavilla: https://www.nginx.com/resources/glossary/http/. Haettu 19/4/2016.

27 Elsen Christian. HTTP/2 is here! Goodbye SPDY? Not quite yet saatavilla: https://blog.cloudflare.com/introducing-http2/. Haettu 19/4/2016. 2015.

28 Dorfman Justin. The Shift from SPDY to HTTP/2, and What It Means for You saatavilla: https://www.maxcdn.com/blog/spdy-http2-shift/. Haettu 23/4/2016. 2015.

29 A Simple Performance Comparison of HTTPS, SPDY and HTTP/2 saatavilla: https://blog.httpwatch.com/2015/01/16/a-simple-performance-comparison-of-https-spdy-and-http2/. Haettu 23/4/2016. 2015.

30 HTTP Compression saatavilla: https://www.httpwatch.com/httpgallery/compression/. Haettu 23/4/2016.