TARTU ÜLIKOOL
MATEMAATIKA-INFORMAATIKATEADUSKOND
Arvutiteaduse instituut
Informaatika eriala

Siim Annuk

Peer-to-peer arhitektuuri rakendamine virtuaalmaailmades

Bakalaureusetöö (6 EAP)

Juhendajad: Dan Bogdanov
Eero Vainikko

Autor: ..... "....." mai 2012
Juhendaja: ..... "....." mai 2012

Lubada kaitsmisele
Professor: ..... "....." mai 2012

TARTU 2012

Sisukord

1 Sissejuhatus

1.1 Mis on võrgupõhised virtuaalmaailmad?

Virtuaalmaailma all mõistetakse tavaliselt arvutil simuleeritud virtuaalset keskkonda, mis võimaldab kasutajatel omavahel suhelda, ringi liikuda ja keskkonnaga interakteeruda. Tavaliselt seostuvad virtuaalmaailmadega kahe- ja kolmemõõtmelised virtuaalsed keskkonnad, kus kasutajad loovad omale avatari, mis neid antud maailmas kujutab ja esindab.

Virtuaalmaailmad erinevad teistest keskkondadest ja mängudest selle poolest, et maailma olek peab säilima isegi siis, kui kasutaja virtuaalmaailmast väljub. Samuti peavad säilima ka kasutaja tehtud muudatused sellele maailmale. Tavaliselt osaleb virtuaalmaailmas suurel hulgal kasutajaid ning kui antud maailm on loodud mängulistel eesmärkidel, siis moodustub nn. *massively multiplayer online role playing game* ehk MMORPG, millest populaarseimad on *World of Warcraft*[1] ja *EverQuest II*[2]. Sellised mängud põhinevad enamasti fantaasia või ulme žanritel, kus mängija ülesandeks on oma tegelast arendada ja ette võtta missioone. Nii nagu reaalses elus, on ka virtuaalmaailmades kasutajate omavaheline suhtlemine küllaltki oluline. Paljud tegevused eeldavad suhtlemist, olgu selleks siis kaaslastega mingiks missiooniks ette valmistumine, mitmekesi mõne ehitise rajamine või ka lihtsalt uute tutvuste loomine. Paljud virtuaalmaailmad võimaldavad kasutajatel suhelda nii tekstipõhiselt kui ka reaalajas heli ja video vahendusel.

Virtuaalmaailmadel on lisaks meelelahutusele ka väga palju muid rakendusi, näiteks e-õpe, mille puhul kasutajateks on õpilased ja õppejõud. Sellised rakendused võimaldavad pidada reaalajas loenguid või õpisessioone, kaasates inimesi samaaegselt üle maailma. Samuti võivad virtuaalmaailmad pakkuda päris elule alternatiivreaalsust, kus saab reaalse eluga sarnaseid tegevusi teha, pakkudes samal ajal päris eluga võrreldes rohkem võimalusi ja suuremat variatsiooni. Lisaks kasutavad paljud ettevõtted virtuaalmaailmu ka oma toodete ja teenuste reklaamimiseks. Üks tuntumaid virtuaalmaailmu on *SecondLife*[3], mis on rakendust leidnud kõigil eelnimetatud eesmärkidel.

1.2 Virtuaalmaailmade ajalugu

Virtuaalmaailma mõiste on vanem kui arvutid, kuna juba Vana-Rooma loodusteadlane Gaius Plinius tegeles tajutava illusiooni uurimisega [1]. Arvutite peal baseeruvad virtuaalmaailmad ja -reaalsused eksisteerivad juba viimased 40 aastat. Üheks esimeseks virtuaalreaalsuse pioneeriks oli Ivan Sutherland ja tema *Ultimate Display* idee, mida mees kirjeldas 1965. aasta kirjutises kui ruumi „milles arvuti saab kontrollida mateeriat“ ning luua *virtuaalmaailmu* [2]. Esimeseks kolmemõõtmeliseks mitme kasutajaga tulistamismänguks oli *Maze War*, mis loodi 1970. aastatel.

1 World of Warcraft, http://www.battle.net/wow [03.03.2012].
2 Everquest II, http://www.everquest2.com [03.03.2012].
3 SecondLife, http://www.secondlife.com [03.03.2012].

Maze War jooksmas Imlac PDS-1 miniarvutil [3].

Virtuaalmaailmad tekkisid 1970. aastatel, kui loodi esimene *Multi-User Dungeon* ehk MUD, mis kujutas endast maailma, kus kasutajad said reaalajas interakteeruda, kasutades algselt tekstipõhiseid käsklusi, ning lugeda keskkonna või teiste kasutajate kirjeldusi.

Üheks esimeseks internetipõhiseks kolmemõõtmeliseks maailmaks oli 1996. aastal Helsingis käivitatud projekt nimega Helsinki Arena 2000[4], mis püüdis kaardistada linna modernseid ja ajaloolisi osasid.

Interneti ja tehnoloogia arenguga täiustuvad ka virtuaalmaailmad ning sarnaselt paljudele teistele rakendustele kolivad ka need üha rohkem kasutajate veebibrauseritesse. Tänu brauseripõhiste mängude võidukäigule ning uute veebistandardite tekkimisele on üha enam kättesaadavaks muutumas kolmemõõtmelised rakendused, mis töötavad otse veebilehe sees.

Ka mobiilseadmetele võivad virtuaalmaailmad tulevikus kättesaadavad olla, sest juba praegu on keskmine nutitelefon võimsuselt võrdne umbes 10 aasta taguse lauaarvutiga ning kolmemõõtmelise graafikaga rakendused nutitelefonidele on juba saadaval.

1.3 Ülesandepüstitus

Klassikaliste klient-server arhitektuuril põhinevate virtuaalmaailmade üheks nõrgaks kohaks on serverid, milledesse virtuaalmaailma kliendid ühenduvad. Kui tegemist on suure kasutajaskonna ja suure kasutatavusega virtuaalmaailmaga, siis kipuvad serverite halduskulud kasvama väga kiiresti ning nende kesise haldamise korral võivad kasutajad kogeda ebameeldivusi antud teenuse tõrkumise tõttu. Lisaks sellele on ka kontroll antud

4 Helsinki Arena 2000, http://www.virtualhelsinki.net [11.03.2012].

keskkonna üle serverite haldajatel, mistõttu võivad viimased ette teatamata oma teenused sulgeda. Klient-server arhitektuuril põhinevad populaarsetest teenustest nii *World of Warcraft* kui ka *Second Life.*

Peer-to-peer (P2P) arhitektuur erineb klient-server arhitektuurist selle poolest, et iga osaleja P2P-l põhinevas võrgus võib käituda nii kliendi kui ka serverina – puudub keskne ühenduspunkt. *Peer-to-peer* tehnoloogial põhinevad näiteks populaarne failivahetuskeskkond *BitTorrent*[5] ja suhtluskeskkond *Skype*[6].

Tuhandete kasutajatega ja rikkaliku sisuga virtuaalmaailmad võivad vajada võrdlemisi palju ressursse, sh. arvutusvõimsust, internetiühenduse ribalaiust ja andmete salvestusmahtu. Klient-server mudeli puhul on pudelikaelaks serverid, milledesse kliendid ühenduvad – serverid peavad olema piisavalt võimsad ning neid peab olema piisavalt palju, et kliente samaaegselt hallata ja virtuaalmaailma simulatsiooni töös hoida.

Peer-to-peer mudel kõrvaldab selle pudelikaela, võimaldades ühel kliendil jagada ülesandeid laiali teistele, vähem koormatud klientidele. Näiteks, kui kasutajad kirjutavad skripte, mis vajavad arvestatava hulga arvutusvõimsust (animeerides mingit objekti või lahendades mõnda arvutusülesannet), siis klient-server mudeli puhul on serveri ülesandeks klientide skripte jooksutada ning suur klientide ja skriptide arv võib serveri üle koormata. Seevastu P2P arhitektuuri puhul on võimalik rakendada koormusjaotusalgoritme ning leida kliendid, kellele võiks delegeerida mõnede skriptide jooksutamise selle alusel, kui palju ressursse ja arvutusvõimsust neil parasjagu üle on.

Selline *peer-to-peer* tehnoloogial põhinev koormuse jaotamine võimaldab pakkuda keerulisemat ja rikkalikumat virtuaalmaailma kogemust klientidele. Koormuse jaotamine võimaldab vähendada teenusepakkuja kulusid interneti teenusele ja energiale. Siiski võib P2P võrgu osaliste koondkulu olla suuremgi kui tavalise klient-server võrgu puhul, kuna kõik osalised võivad olla võrdselt hõivatud, andes oma panuse ressursside kasutamisse. Samal ajal lisab P2P kasutamine ka turvariske, võrreldes klient-server mudeliga, milles on sisuliselt ainult üks usaldust vajav komponent – server.

Käesoleva töö eesmärgiks on võrrelda klient-server mudeli ja *peer-to-peer* mudeli efektiivsust ja P2P arhitektuuri rakendatavust virtuaalmaailmades. Selleks viisime läbi katsed teadusuuringute tarbeks ehitatud virtuaalmaailmaga *VirtualLife*.

5 BitTorrent, http://www.bittorrent.com [11.03.2012].
6 Skype, http://www.skype.com [20.03.2012].

1.4 Kirjanduse ülevaade

Enamus uurimistöid *peer-to-peer* arhitektuuril põhinevatest tehnoloogiatest keskenduvad peamiselt failivahetuse ja voogesituse valdkondadele. Huvi *peer-to-peer* tehnoloogiate virtuaalmaailmadele rakendatavuse vastu on tõusmas ning tekkinud on mitmeid projekte, mis antud tehnoloogiat rakendada püüavad. Järgnevalt toome välja mõned olulisemad virtuaalmaailmade projektid, mis kasutavad *peer-to-peer* arhitektuuri.

1.4.1 Solipsis

Solipsis [4] on tasuta ja avatud lähtekoodiga projekt, mis püüab luua piiramatu suurusega kolmemõõtmelist virtuaalmaailma, milles saaks osaleda piiramatul hulgal kasutajaid. Projekti üheks kesksekseesmärgiks on eemaldada vajadus tsentraalsete serverite järele, et niiviisi anda kasutajatele rohkem vabadust erinevate virtuaalmaailma segmentide loomisel.

Iga osaline on otseühenduses teiste osalistega, kelle avatarid on parajasti selle avatari huvipakkuvas piirkonnas. Suhtlus osalistega, kes on antud avatari huvipakkuvast piirkonnast kaugemal, käib hierarhiliselt läbi RayNet [5] kattevõrgu, mida Solipsis kasutab.

Solipsis hajutab füüsika- ja kokkupõrkearvutused, lastes igal osalisel eraldi arvutada tema asukoha ja füüsikasimulatsiooni arvestades väikese koguse objekte osalise avatari ümbruses.

1.4.2 LifeSocial

LifeSocial [6] on *peer-to-peer* arhitektuuril põhinev teaduslikel eesmärkidel loodud sotsiaalvõrgustiku projekt. Kuigi sotsiaalvõrgustikud ei ole otseselt seotud virtuaalmaailmadega, pakuvad nad siiski sarnaseid teenuseid, nagu kasutajaprofiilid, erinevad suhtlusviisid kasutajate vahel ning koostöövõimalused.

LifeSocial saavutab oma funktsionaalsuse pistikprogrammide abil, näiteks on olemas pistikprogrammid sisse logimiseks, profiili haldamiseks, failide vahetamiseks jne.

Antud projekt implementeerib *peer-to-peer* arhitektuuri kasutades FreePastry DHT[7] (*Distributed Hash Table*) süsteemi, mis on aluseks hajusale süsteemi säilitamisele. Lisaks sellele pakub LifeSocial turvalist infrastruktuuri näiteks autenditud ja konfidentsiaalseks suhtluseks ning juurdepääsukontrolliks.

1.4.3 Donnybrook

Donnybrook [7] on *peer-to-peer* tehnoloogial põhinev teadusprojekt, mille eesmärgiks on luua võrgupõhine süsteem, mis võimaldaks läbi viia suuremahulisi lahingusimulatsioone ilma tsentraliseeritud servereid kasutamata. Võrgu ribalaiuse vähendamine saavutatakse selle läbi, et uuendused saadetakse mängijatele vastavalt nende mängijate huvipakkuvale piirkonnale (*interest set*). See tähendab, et vähemtähtsad oleku uuendused saadetakse harvemini ja seeläbi vähendatakse võrguliikluse mahtu.

7 FreePastry, http://www.freepastry.org [23.03.2012].

Osalised, kellel on rohkem ressursse (näiteks kiirem internetiühendus, suurem arvutusjõudlus), valitakse sõnumi edastajate rolli, et vähemate ressurssidega osalised saaksid nende kaudu oma sõnumeid edastada.

1.4.4 VAST

VAST [8] on avatud lähtekoodiga *peer-to-peer* arhitektuuril põhinev võrguteek, mis on peamiselt suunatud MMO-tüüpi (*massively multiplayer online*) mängudele. Peamine probleem, mida VAST lahendada püüab, on õigete naabrite leidmine, kellega ühenduda, ja konsistentse maailmapildi tagamine. Selle probleemi lahendamiseks kasutatakse Voronoi diagrammidel põhinevat kattevõrku ja „avalda/telli“ (*publish/subscribe*) mehhanismi.

Antud projekti poolt kasutatav „avalda/telli“ mehhanism võimaldab tippudel „tellida“ mingist kahemõõtmelisest piirkonnast uuendusi ja sõnumeid või „avaldada“ teatud piirkonda enda uuendusi ja sõnumeid.

1.5 Töö ülevaade

Käesolev töö koosneb neljast peatükist:

- Esimeses peatükis antakse ülevaade võrgupõhistest virtuaalmaailmadest ja tutvustatakse klient-server ja *peer-to-peer* arhitektuuri.
- Teises peatükis tutvustatakse projekti VirtualLife ning selle juurutust ja arhitektuuri.
- Kolmandas peatükis antakse põhjalikum ülevaade VirtualLife'i komponentidest ning vaadeldakse, kuidas *peer-to-peer* tehnoloogiat erinevates rakenduse kihtides on kasutatud.
- Neljandas peatükis uuritakse P2P tehnikate mõju teenusepakkuja kuludele ja kliendi kasutusmugavusele.

1.6 Autori panus

Bakalaureusetöö autor on töötanud VirtualLife'i arendusmeeskonnas programmeerijana konsortsiumi Eesti partneri Cybernetica AS juures. Autori olulisemad panused käesolevas töös on:

- võrgukihis TLS turvakanali kasutuselevõtt;
- VirtualLife'i võrgu tippude omavaheliste otseühenduste võimaldamine *NAT punchthrough* tehnoloogiat kasutades;
- P2P kõneteenuse integreerimine;
- P2P tehnikate mõju mõõtmise eksperimentide kavandamine ja programmeerimine (koostöös meeskonnaga);
- P2P tehnikate mõju mõõtmise eksperimentide läbiviimine virtualiseeritud eksperimendikeskkonnas ja tulemuste analüüs (koostöös meeskonnaga) ja
- P2P tehnikate mõju mõõtmise eksperimentide läbiviimine füüsilisest masinatest koosnevas keskkonnas ja tulemuste analüüs (iseseisvalt).

2 VirtualLife

2.1 Projekti tutvustus

VirtualLife (VL) on Euroopa Komisjoni 7. raamprogrammi teadusprojekt, mis mille eesmärgiks on luua virtuaalse maailma platvorm, keskendudes turvalisuse, demokraatia ja koostöö aspektidele. VL pakub kolmemõõtmelist keskkonda, mis kombineerib virtuaalse õigussüsteemi, tugeva turvataristu ja *peer-to-peer* arhitektuuri ja on sobilik kasutamiseks hariduses, e-kaubanduses, ettevõtluses ja meelelahutuses.

Projekt algas 2008. aasta jaanuaris ning lõppes 2011. aasta sügisel. Teadus- ja arendustöös olid kaasatud järgmised partnerid:

- Nergal S.r.l. (Itaalia) – töö koordineerimine, Virtuaalse Riigi teenused
- Deep Blue Consulting and Research (Itaalia) – lõppkasutaja kaasamine
- Cybernetica AS (Eesti) – hajussüsteemide arhitektuur ja andmeturve
- Digital Video S.p.A. (Itaalia) – kasutajaliides, skriptimismootor
- Geumacs (Rumeenia) – nõuete analüüs
- Vilnius University (Leedu) – tulemuste levitamine
- Panebarco S.a.s. (Itaalia) – graafika, äriarendus
- Tavae (Prantsusmaa) – graafikamootor, hajussimulatsioon

Konsortsium oli hästi tasakaalustatud ning vastutusalad tasakaalus. Cybernetica vastutas loodavas süsteemis andmevahetuse ja infoturbetehnikate rakendamise eest.

2.2 Juurutus

VirtualLife on *peer-to-peer* tehnoloogial põhinev rakendus, kus iga võrku ühendunud tipp (*node*) võib ühenduda iga teise võrgutipuga. Kuigi vastavalt *peer-to-peer* arhitektuuri tavadele võivad kõik tipud käituda nii serveri kui ka kliendina, on kasulik eristada mõningaid tippe teistest, andes neile rohkem kontrolli ja rohkem teenuseid. Nii moodustub hübriidne *peer-to-peer* võrk, kus mõned kesksed tipud toetavad ülejäänud võrgu tööd. Hübriidsed *peer-to-peer* süsteemid, näiteks Skype, on küllaltki populaarsed.

Iga VL tipp pakub erinevaid teenuseid, näiteks kasutajaliides klientrakendustes, andmebaas andmete salvestamiseks serverrakenduses, koormusjaotus jne. Kõiki teenuseid ei ole mõttekas kõigil tippudel pakkuda ning arvestades erinevate tippude rolli VirtualLife

maailmas, jaotatakse tipud järgmiselt:

1) Klientrakendused *(Virtual Client, VC*)
2) Virtuaalse tsooni serverid *(Virtual Zone, VZ*)
3) Virtuaalse riigi serverid (*Virtual Nation, VN*)

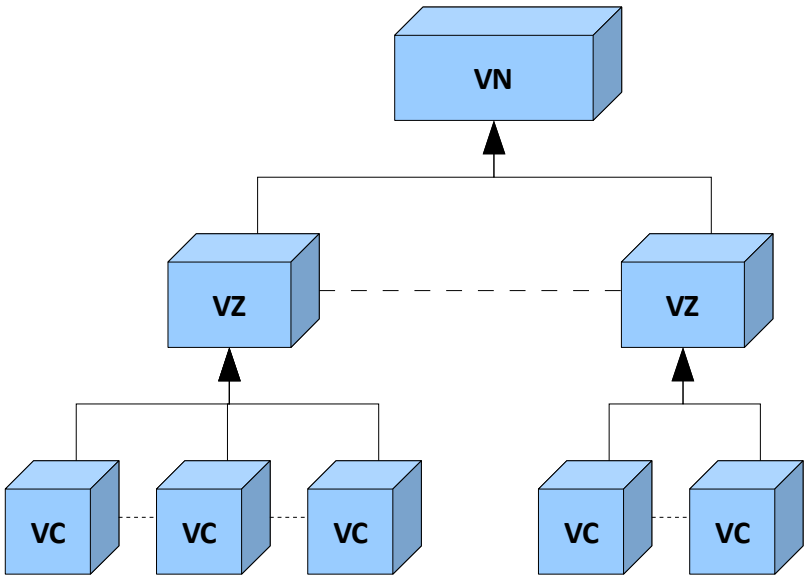


Diagramm 1. VirtualLife võrgu hierarhia.

Peer-to-peer režiimis loovad võrgutipud ka omavahelisi otseühendusi, mida illustreerivad diagrammil 1 katkendjooned.

2.2.1 Klientrakendus

Klientrakendus on lõppkasutaja jaoks mõeldud rakendus, mis kuvab kolmemõõtmelist virtuaalmaailma ning kus kasutaja interakteerub selle maailmaga. Klientrakendus ühendub enamasti virtuaalse riigi serverisse, mille kaudu on omakorda võimalik siseneda virtuaal-setesse tsoonidesse. Klientrakendus võimaldab töötada ka ühenduseta režiimis, ilma VN serverisse ühendumiseta, ning lubab sellisel juhul kasutada väiksemat alamhulka VC funktsionaalsustes. Arendusprotsessi lihtsustamiseks loodi klientrakendusest ka graafilise liideseta käsurearakendus, mille abil kontrolliti mitmete võrguprotokollide tööd. Klient-rakendus pakub järgmiseid teenuseid:

- võrgutippude vaheliste ühenduste loomine;
- virtuaalse riigi serverisse ja tsooni ühendumine;
- kolmemõõtmelise virtuaalse maailma kuvamine ja sellega interakteerumine;
- füüsika simuleerimine;

- skriptide käitamine ja
- teksti- ja helipõhised suhtlusteenused.

2.2.2 Virtuaalse riigi server

Virtuaalse riigi eesmärgiks on eelkõige kasutajate virtuaalmaailma sisse lubamine ning virtuaalsete tsoonide haldus. Iga VZ, mis soovib antud VN-i kuuluda, peab aktsepteerima selle VN-i reegleid (näiteks virtuaalsed seadused või teistsugused füüsikaseadused). Ühes virtuaalses riigis võib eksisteerida korraga mitu VZ-i. Lisaks sellele pakub virtuaalse riigi server ka muid teenuseid:

- kasutajate identiteetide haldamine;
- kasutajate sertifikaatide signeerimine ja valideerimine;
- identiteedipõhise kattevõrgu haldamine ja
- identiteedipõhiste ühenduste võimaldamine.

2.2.3 Virtuaalse tsooni server

Tsoon on VirtualLife'i virtuaalse maailma osa. Tsooniserveri ülesanneteks on graafika-spetsiifiliste ressursside (tekstuurid ja geomeetria), gruppide ja õiguste haldamine ning virtuaalreaalsuse mootori kattevõrgu ja tsoonidevahelise esemevahetuse võrgu haldamine. Lisaks sellele vastutab VZ virtuaalmaailma oleku püsimise eest – kasutajate loodud objektid ja muudatused maailmale peavad jääma püsima isegi VZ-i taaskäivitamise järel. Virtuaalse tsooni server pakub muuhulgas järgmisi teenuseid:

- kolmemõõtmelise maailma simulatsiooni haldamine;
- tekstipõhine suhtlus tsoonis olevate kasutajate vahel ja
- maailmas sisalduvate ressursside haldamine.

2.3 Rakenduse arhitektuur

VirtualLife koosneb kolmest peamisest rakendusest – klientrakendus, virtuaalse riigi server ja virtuaalse tsooni server. Kõik rakendused kasutavad sarnast arhitektuuri ning on üles ehitatud järgmiste teekide abil:

vlcommon – üldised andmestruktuurid ja üldkasulikud teenused, näiteks:

- logimissüsteem;
- identiteediga seotud andmestruktuurid *Guid* ja *Identity;*
- süsteemi monitooringu teenused ja
- konfiguratsioonifailide haldamine.

vlsec – turvalisuse ja krüptograafiaga seotud teenused ja funktsionaalsus:

- X.509 sertifikaatide loomine ja käitlemine;
- digiallkirjastamine;
- juurdepääsukontrolli poliitika ja
- krüptograafilised primitiivid (sümmeetriline šiffer, avaliku võtme šiffer, räsialgoritmid, sõnumite autentimise algoritmid).

vlnet – võrgutippude haldamine läbi kattevõrgu ning sõnumite edastamine ja vastu võtmine voogude abil. Teegi peamisteks ülesanneteks on:

- sissetulevate ühenduste kuulamine ja uute ühenduste loomine;
- teenuste ülesseadmine ja haldamine;
- andmevoogude haldamine;
- võrgustatistika kogumine;
- *NAT punchthrough* teenuse pakkumine ja kasutamine ühenduste loomisel ja
- autenditud ja krüpteeritud TLS sidekanali loomine ning haldamine.

vlprotocol – see teek sisaldab peamiselt VirtuaLife rakenduse spetsiifilisi teenuseid ja funktsionaalsust. Selle teegi ülesanneteks on muuhulgas pakkuda järgmiste teenuste kliendi poole komponente:

- teksti- ja helipõhised suhtlusteenused;
- lepingute sõlmimise teenuste süsteem;
- teadete edastamise süsteem ja
- hääletussüsteem.

vlscript – skriptimismootor, mis võimaldab kasutajatel kirjutada erinevaid skripte nt. mingi objekti animeerimiseks virtuaalses maailmas või matemaatiliste võrrandite lahendamine jne. Skriptimismootori ülesanneteks on:

- kapseldada Lua[8] skriptimismootorit;
- siduda objekte Lua keskkonna ja VirtualLife keskkonna vahel;
- pakkuda teenuseid vektoralgebra, animeerimise jms. jaoks ja

8 Lua Programming Language, http://www.lua.org [13.04.2012].

- hallata skriptide persistentsust ja omavahelist kommunikatsiooni.

vrengine – madalama taseme teek, mis haldab kogu VirtualLife'i kolmemõõtmelist virtuaalreaalsust. Teegi peamisteks ülesanneteks on muuhulgas:

- kolmemõõtmelise simulatsiooni teostamine ja pildi kuvamine;
- sõnumite korrektne ja optimaalne suunamine virtuaalkeskkonnas;
- üksuste (e*ntity*) persistentsuse haldamine ja
- virtuaalse riigi ja tsooni seaduste propageerimise haldamine.

3 *Peer-to-peer* tehnoloogia VirtualLife'is

3.1 Sõnumivahetuskiht

Iga VL võrgu tipp saab teise tipu külge ühenduda nii IP aadressi kui ka VirtualLife Guid (*Globally Unique Identifier*) abil. Kuna virtuaalse riigi server haldab ka kõikide registreeritud kasutajate (tippude) andmeid ning teab iga kasutaja viimase sisselogimise andmeid (sh. IP aadressi) ning Guidi, siis on Guidi abil ühendumisel võimalik vastav IP aadress virtuaalse riigi serverist küsida.

3.1.1 Tippudevahelised otseühendused

Tihti asub ühe ruuteri taga mitu arvutit, mis moodustavad lokaalsete IP aadressidega kohtvõrgu. Need lokaalsed IP aadressid aga ei ole sellest kohtvõrgust väljaspool (internetis) üheselt määratud ning seetõttu peab marsruuter need lokaalsed aadressid ümber transleerima globaalseks väliseks IP aadressiks. Sellist protsessi nimetakse *Network Address Translation* ehk NAT.

Otseühenduse loomine ei pruugi töötada, kui üks tippudest on NAT serveri taga. Sellisel juhul saavad tipud käivitada *NAT punchthrough* protokolli, mis ühendab mõlemad tipud ühe ja sama serveri külge, mis ei tee NAT-i. Seejärel leiab see protokoll mõlema tipu avaliku IP aadressi ja pordi, tekitades samal ajal ka NAT-itud ruuterisse vajalikud vastavused. Leitud aadressid saadetakse vastavalt kummalegi tipule ning seejärel on otseühendumine võimalik.

Kuna võimalikke võrgukonfiguratsioone on palju, siis *NAT punchthrough* mehhanism ei ole täielikult töökindel. Serverite tulemüürid võivad olla väga ranged ja nii võib *NAT punchthrough* kahe erinevas võrgus oleva tipu vahel ebaõnnestuda. VirtualLife'i võrgukiht püüab seda probleemi lahendada, saates ühendatavale tipule sõnumi läbi virtuaalse tsooni serveri, mille järel luuakse tagasiühendus ühendust alustanud tipule. Kui ka tagasiühendus ebaõnnestub, siis P2P ühenduse loomine antud tippude vahel ei ole võimalik ning sõnumid saadetakse antud tippude vahel läbi kolmanda tipu, milleks on enamasti sobiv virtuaalse tsooni server.

3.1.2 Sõnumite krüpteerimine

VirtualLife'i võrgukiht võimaldab väljuvaid sõnumeid krüpteerida ning krüpteeritud sõnumeid vastu võtta. Sõnumite saatmisel on võimalik määrata, kas väljuv sõnum krüpteeritakse või mitte. Suuremahulisemad ja vähemtähtsad andmed on enamasti krüpteerimata (nt. objektide olekuinformatsioon). Tähtsamad andmed, sealhulgas kasutajaga seotud isiklik informatsioon ja privaatvestlused, on aga krüpteeritud. Krüpteeritud sõnumite vahetamiseks kahe tipu vahel luuakse uus spetsiaalne turvakanal (andmevoog), mille sisse

tehakse krüpteeritud TLS[9] tunnel. Turvakanali loomisel käivitatakse poolte vahel kõige pealt nn. käepigistusprotsess (*handshake*) ning seejärel on võimalik andmeid konfidentsiaalselt vahetada.

Turvakanali rakendamisest VirtualLife'i keskkonnas on koostatud magistritöö [9] ja avaldatud artikkel [10].

3.2 Virtuaalreaalsuse mootor

Virtuaalreaalsuse mootor (*vrengine*) vastutab kogu virtuaalse maailma oleku eest ning tema peamisteks ülesanneteks on:

- kolmemõõtmelise graafika renderdamine;
- objektide haldamine;
- füüsikasimulatsiooni käitamine;
- rakenduse tasemel suhtluskanalite loomine ja haldamine erinevate võrgutippude vahel;
- virtuaalse maailma olekuinformatsiooni pakendamine ning saatmine teistele võrgutippudele ja
- sõnumite suunamine õigetele saajatele.

3.2.1 Sõnumid

Sõnumite vorming on virtuaalreaalsuse mootoris konkreetselt määratud ning sisaldab vajalikul hulgal informatsiooni sõnumi allika, sihtkoha ja parameetrite kohta. Lisaks sellele on sõnumid kategoriseeritud kolme gruppi – kõrge ja madala sagedusega ning hooldussõnumid. Kõrge sagedusega sõnumid on näiteks *UPDATE*-tüüpi sõnumid (st. sõnumid, mille sisuks on mingi kindla objekti või andmehulga värskendamine, näiteks liikuva objekti positsioon maailmas), mida saadakse vähemalt kord kaadris ainult otseühendunud tippudele, kes ka antud sõnumist on huvitatud. Madala sagedusega sõnumid sisaldavad näiteks *PERMISSION*-tüüpi sõnumeid, mis sisaldavad informatsiooni mingi objekti õiguste kohta ja mida saadetakse harva. Hooldussõnumid sisaldavad muuhulgas sõnumeid, mis hoiavad kogu kattevõrgu olekut värskena.

3.2.2 Vastutuse süsteem

Üheks väga oluliseks aspektiks virtuaalreaalsuse mootori ja terve VirtualLife'i rakenduse juures on vastutuse (*authority*) süsteem. Kui kaks erinevat võrgutippu juhtuvad sama-

9 Transpordikihi turbeprotokoll (ingl. Transport Layer Security) krüpteerib võrguühenduse segmente suhtluskanali otspunktides.

aegselt muutma ühte kindlat objekti, võivad tekkida ebakorrapärasused virtuaalse maailma simulatsioonis. Näiteks, kui kasutajad A ja B tahavad korraga liigutada palli C erinevates suundades, siis palli lõplik asukoht ei pruugi kummagi kasutaja jaoks olla soovitud asukoht. Vastutuse süsteem aitab seda probleemi vältida, andes igale objektile ühe kindla omaniku, kes võib antud objekti omadusi muuta.

Enamasti saab uue objekti omanikuks selle looja ning hiljem on võimalik volitusi ka edasi anda. Kui kasutaja väljub maailmast, siis liiguvad temale kuuluvate objektide volitused virtuaalse tsooni serverile ning maailma sisenedes jälle kasutajale tagasi. Kui kasutaja soovib muuta mingi üksuse omadusi, millele tal volitusi pole, siis saab kasutaja lokaalne virtuaalse reaalsuse mootor need volitused laenata üksuse omanikult. Tehtud muudatused propageeritakse sellisel juhul üle võrgu ka teistele kasutajatele, kes antud üksust näevad. Et vältida teenusetõkestamise rünnet, st. olukorda, kus näiteks üks kasutaja omandab kõikide üksuste õigused igaveseks mingis tsoonis, on laenule andmise aeg piiratud ning selle aja möödumisel lähevad objekti õigused automaatselt tagasi algsele omanikule.

3.2.3 Vaatlejale huvipakkuv piirkond

Selleks, et vähendada võrgukoormust, kasutab virtuaalreaalsuse mootor nn. huvipakkuva piirkonna (*area of interest*) metoodikat. Selle metoodika eesmärgiks on piirata andmehulka, mida võrgutippude vahel saadetakse. Väga suure osa VirtualLife võrguliiklusest hõivavad UPDATE-tüüpi sõnumid, mis on enamasti küll väiksemahulised, kuid mida saadetakse väga tihti – umbes kord kaadris. Sellisteks sõnumiteks on näiteks objektide asukoha ja asendi värskendamise sõnumid.

Virtuaalse reaalsuse mootor kasutab küll ennustamist (näiteks liikuva objekti trajektoori määramist ilma välisvõrgust tuleva informatsioonita), kuid halvimal juhul tuleb saata üks UPDATE-tüüpi sõnum kaadri kohta. Et aga veelgi efektiivsemalt kõrgsageduslikke sõnumeid saata, arvestab virtuaalse reaalsuse mootor, millised piirkonnad võiksid saada rohkem tähelepanu ja millised vähem. Näiteks kasutaja vaateväljale lähedal asuvad objektid peaksid saama võimalikult tihti uuendusi, kuid vaateväljast eemal olevad objektid harvem või üldsegi mitte.

3.3 Füüsikasimulatsioon

VirtualLife võimaldab füüsikasimulatsiooni arvutusi teha hajusalt, hoides seejuures võrgukoormuse minimaalsena. Füüsiline objekt võib virtuaalses füüsilises maailmas eksisteerida ühes kolmest võimalikust vormist:

- Staatiline objekt on ilma massita objekt, mis ei saa liikuda, kuid saab põrkuda teiste objektidega. Staatilisteks objektideks on näiteks maailma geomeetriline esitus.

- Kinemaatiline objekt on samuti massita ning seda on võimalik animeerida kasutaja poolt. Kinemaatiline objekt saab küll põrkuda teiste objektidega, kuid teised objektid ei mõjuta kinemaatilist objekti. Kinemaatiliseks objektiks on näiteks kasutaja avatar.

- Dünaamiline objekt on positiivse massiga objekt ning on mõjutatud füüsilise maailma poolt. Kasutaja ei saa seda objekti liigutada. Dünaamiliseks objektiks on näiteks piljardikuul.

Enamus füüsikamootoreid on ennustavad, st. et võttes aluseks mingi oleku ajahetkel T_0, arvutatakse mingi uus olek ajahetkel T_n. Kui sellist meetodit kasutada hajussüsteemis igal tipul, siis tänu arvutite erinevusele, ujukomaarvutuste ebatäpsusele ja muudele teguritele ei ole võimalik tagada järjekindlaid maailma olekuid süsteemis olevate tippude vahel.

Üks võimalus füüsikalise maailma järjekindluse hoidmiseks on valida üks võrgutippudest füüsika arvutuste eest vastutavaks, sarnaselt klassikalisele klient-server mudelile. Sellisel juhul peab füüsikat serveeriv tipp arvutama füüsikasimulatsiooni olekud ning need siis propageerima kõikidele teistele tippudele. Sellise lähenemise miinuseks on latentsus, mis tekib kasutaja sisendi ja füüsikalise maailma olekute muutumisel vahel. Lisaks sellele peaks füüsikaserver arvutama füüsikaolekud iga tipu jaoks, mis lisab tunduvalt koormust serverile.

VirtuaLife füüsikamootor eeldab, et võrgutipud suudavad füüsikasimulatsiooni lokaalselt arvutada. Iga tipp aga arvutab ainult nende objektide olekuid, mis asuvad temale huvipakkuvas füüsilises piirkonnas. See piirkond on enamasti määratud mingi kolmemõõtmelise raadiusega antud tipu asukohast virtuaalses füüsikalises maailmas. Selline lähenemine võimaldab vähendada võrguliiklust, sest objektide, mis asuvad väljaspool antud huvipiirkonda, olekuid ei pea teistele tippudele saatma.

3.4 Skriptimine

Kasutajatel on võimalik luua skripte, kasutades Lua skriptimiskeelt. Skriptid võimaldavad kasutada suurt hulka VirtualLife'i funktsionaalsusest reaalajas, näiteks luua ja animeerida erinevaid objekte, mänge jpm. Skriptide käitamise ja haldamise eest vastutab vlscript teegis realiseeritud skriptimismootor.

Skript töötab sellel VirtualLife võrgu tipul, kus ta käivitati, kuid võib töö käigus migreeruda ka mõnele teisele tipule. Näiteks, kui skripti loonud ja käivitanud kasutaja logib maailmast välja, migreeritakse skript käitamiseks virtuaalse tsooni serverile. Kasutaja taaslogimisel maailma migreeritakse skript jälle tagasi kasutaja võrgutipule.

Kui kasutaja loob palju ressursinõudlikke skripte, siis skriptimismootor võib ülekoormuse vältimiseks ka skripte teistele võrgutippudele migreerida. Skripti migreerimine on võimalik juhul, kui tipp on üle koormatud (vastavalt mõõdetud süsteemimeetrikatele), tipuga on ühenduses vähemalt üks vähem koormatud tipp ning antud tipus leidub skripte, mida on võimalik migreerida. Skriptid, mida ei saa migreerida, on näiteks:

- skriptid, mis töötavad kõikides tippudes, sest see ei muudaks tippude koormust;
- skriptid, mis interakteeruvad kasutajaga ning jooksevad kasutaja tipus;
- skriptid, mis on äsja migreerunud, ei tohi koheselt uuesti migreeruda ja

- skriptid, mis moodustavad lõviosa tipu koormusest, ei tohiks migreeruda, sest see tõenäoliselt koormaks üle teise tipu.

Kui skriptimismootor avastab ülekoormuse, migreerimise tingimustele vastava skripti ja migreerimiseks sobiliku siht-tipu, siis skripti töö antud tipus lõpetatakse ning skripti volitused antakse üle siht-tipule. Skripti sisemised olekud saadetakse siht-tipule, kus olekud taastatakse ning skripti töö jätkub.

Lisaks migreerimisele on skriptimismootor üles ehitatud nii, et kasutajatel ei oleks võimalik koostada skripte, mis kunagi tööd ei lõpeta või sisaldaksid lõpmatuid tsükleid. Iga kaadri jooksul antakse skriptidele mingi kindel maksimaalne aeg töötamiseks ning selle ületamisel jätkatakse järgmisel kaadril pooleli jäänud kohast. See aitab võidelda kasutajate pahatahtlikkuse vastu.

4 *Peer-to-peer* tehnoloogia efektiivsus

4.1 Eksperimentide läbiviimine ja mõõtmismetoodika

4.1.1 Sissejuhatus

VirtualLife projekti lõppfaasis viidi läbi eksperimendid mõõtmaks, millist mõju avaldab *peer-to-peer* mudeli kasutamine antud rakendusele võrreldes klient-server mudeliga. Eksperimentide peamisteks eesmärkideks oli leida erinevusi VirtualLife platvormi jõudluses ja kogukuludes.

Eksperimentide läbiviimiseks ja analüüsitavate andmete kogumiseks lisati VirtuaLife'i võimalus rakendust kompileerida ja käivitada *peer-to-peer* või klient-server režiimis. Lisaks sellele loodi süsteem testide automaatseks käitamiseks – iga võrgutipu klient-rakendus oli võimalik konfigureerida nii, et käivitamisel logis rakendus automaatselt vastavasse virtuaalse tsooni serverisse ning käivitas mingi kliendipoolse skripti. Selline süsteem tagas piisavalt suure determinismi testide korduva käivitamise suhtes.

Iga eksperimendi käigus logiti eksperimendile sobiliku intervalliga järgmised andmed:

- virtuaalses maailmas eksisteerivate objektide asukohad ja orientatsioonid;
- süsteemimeetrikad, sh. protsessori kasutus, mälu kasutus, võrgu kasutus jne. ja
- virtuaalreaalsuse mootori parameetrid.

Andmete võrdlemiseks ja analüüsi lihtsustamiseks logiti andmed millisekundilise täpsusega ajatempliga.

4.1.2 Eksperimentide läbiviimine

Automatiseeritud eksperimendid viidi läbi kuni 15 tipuga HPC klasterarvutil. Igal tipul oli 24 tuuma (protsessoriks AMD Opteron 6168) ja 48 GB mälu. Väiksemate virtuaalmasinate loomiseks kasutati OpenVZ[10] virtualiseerimistehnoloogiat.

Kuigi VirtualLife klientrakendus eeldab tänapäevase graafikakaardi olemasolu, siis virtualiseerimistehnoloogiad ei ole veel jõudnud riistvaraliste graafikakiirendite valdkonda, mistõttu tuli graafikaga seotud operatsioonid teha tarkvaralise emuleerimise teel, kasutades Mesa[11] teeki. See omakorda aga tekitas mõningaid jõudluse probleeme klientrakenduses ning olukorra leevendamiseks lisati VirtualLife'i võimalus graafika renderdamine klient-rakenduses välja lülitada.

10 OpenVZ, http://openvz.org [13.04.2012].
11 The Mesa 3D Graphics Library, http://www.mesa3d.org [14.04.2012].

Lisaks kirjeldatud virtualiseeritud keskkonnale korrati käesoleva töö raames mõningaid eksperimente ka füüsilistest lauaarvutitest koosnevas keskkonnas, kuna virtualiseeritud testkeskkond võis nende eksperimentide tulemusi mõjutada.

Üheks probleemiks eksperimentide tulemuste analüüsimisel osutus andmete joondamine, kuna iga klientrakenduse logi ajatemplid olid logitud relatiivselt klientrakenduse käivitamise suhtes, st. logide ajatemplid algasid nullist ning suurenesid etteantud intervalliga. Korratud eksperimentide jaoks muudeti ajatemplid absoluutseteks, st. aeg millisekundites alates Unixi epohhist (1. jaanuar 1970). Absoluutsed ajatemplid võimaldasid logide sissekandeid võrrelda täpselt, ilma vajaduseta andmeid joondada.

Kordusekspериmentide läbiviimiseks kasutati 16 lauaarvutit, mis olid ühendatud kiiresse kohtvõrku. Arvutitel olid kahetuumalised Intel Core Duo E4600 (2.4 GHz) protsessorid ja 2 GB operatiivmälu ning operatsioonisüsteemiks oli OpenSUSE 11.0.

Iga eksperimendi ettevalmistamiseks virtualiseeritud testkeskkonnas kopeeriti vastav *peer-to-peer* või klient-server installatsioon igale klasterarvuti tipule *rsync*[12] abil, kirjutades üle võimalikud eelmisest eksperimendist tekkinud muudatused. Eksperimentide käivitamiseks kasutati *parallel-ssh*[13] tööriista, mis võimaldab saata süsteemikäske üle SSH[14] mitmele arvutile samaaegselt.

Eksperimendi läbiviimise protsess oli järgmine:

1) koostada kliendipoolne skript eksperimendi detailidega (näiteks liigutada kasutaja avatari mingil trajektoori määratud aja jooksul);

2) käivitada virtuaalse riigi server;

3) käivitada virtuaalse tsooni server;

4) käivitada vajalikud kliendid koos vastava skriptiga;

5) kui skript on töö lõpetanud, peatada kliendid ja serverid;

6) koguda kokku logid klientidest ja serveritest ning

7) analüüsida logisid ja uurida hüpoteesi paikapidavust.

4.2 Hüpoteesid

Peer-to-peer tehnoloogia efektiivsuse mõõtmiseks VirtualLife's tuli uurida, kuidas P2P mõjutab kasutaja kogemust (*user experience*) ja platvormi haldamise kogukulusid (*total cost of ownership*). Selleks pakuti välja järgmised hüpoteesid:

1) Kasutaja kogemus

12 rsync, http://gcc.gnu.org/rsync.html [14.04.2012].
13 Parallel SSH, http://www.theether.org/pssh [14.04.2012].
14 Secure Shell, http://en.wikipedia.org/wiki/Secure_Shell [14.04.2012].

a) **Reageerimisaeg säilib ka rikkalikus keskkonnas.** Klientrakenduse reageerimisaeg sõltub sõnumite transpordiajast ning avataride ja objektide arvust virtuaalses maailmas.

b) **Korrektsus säilib ka rikkalikus keskkonnas.** Korrektsus on virtuaalse maailma konsistentsuse mõõt vastavalt objektide ja avataride arvule. Konsistetsuse all mõistame seda, kas objektid asuvad õiges kohas õigel ajal.

c) **Robustsus säilib ka rikkalikus keskkonnas.** Robustsus on siin konsistentsuse ja sõnumite edastusaja mõõt, kui võrgus ilmnevad jõudlusmuutused.

2) Platvormi abil teenuse pakkumise kogukulud

a) **Ribalaiuse kasutus on normi piirides:** ribalaiuse kogukulu virtuaalse tsooni serveri jooksutamisel on *peer-to-peer* lähenemist kasutades väiksem kui klient-server lähenemise puhul.

b) **Protsessori kasutus on normi piirides:** tsooni serverite protsessori kogukasutus on *peer-to-peer* lähenemist kasutades väiksem kui klient-server lähenemise puhul.

4.3 Eksperimendid

Kokku tehti 7 eksperimenti *peer-to-peer* ja klient-server režiimis. Nendeks eksperimentideks olid:

1) Ühendunud klientide arvu mõju platvormi haldamise kogukuludele.

2) Kasutajate skriptide koormusjaotuse mõju hajussimulatsioonile.

3) Virtuaalse maailma konsistentsus objektide loomise suhtes.

4) Virtuaalse maailma konsistentsus objektide asukohtade värskendamise suhtes.

5) Võrgu ribalaiuse varieerumise mõju konsistentsusele objektide asukohtade värskendamise suhtes.

6) Võrgu latentsuse varieerumise mõju konsistentsusele objektide asukohtade värskendamise suhtes.

7) Kasutajate skriptide koormusjaotus heterogeense võimsusega arvutivõrgus.

Iga järgneva alapeatüki all toome eksperimendi hüpoteesi, kirjelduse ja kogutud andmed ning analüüsi tulemused. Eksperimentide kirjeldused, tulemuste analüüsid ning joonised põhinevad VirtualLife projekti tarnel [11], mille koostamisel ka autor osales.

4.3.1 Eksperiment 1: Ühendunud klientide arvu mõju platvormi haldamise kogukuludele

Hüpotees

Peer-to-peer režiimis kasutab virtuaalse tsooni server vähem ressursse ja skaleerub paremini ühendunud klientide arvu suhtes kui klient-server režiimis.

Eksperimendi kirjeldus

Eksperimendi läbiviimiseks kasutati kahest füüsilisest arvutist koosnevat klasterarvutit 16 virtuaalmasinaga. 15 virtuaalmasinat kasutati klientrakenduste jooksutamiseks ja ühte virtuaalse riigi serveri ja virtuaalse tsooni serveri jooksutamiseks.

Eksperiment koosnes järgmistest sammudest:

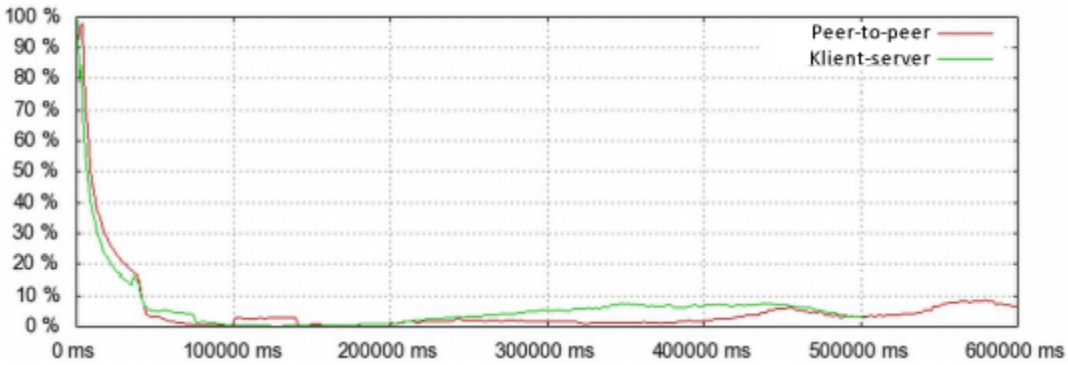
1) virtuaalse riigi ja virtuaalse tsooni serverid käivitati;
2) 15 klientrakendust käivitati 10 sekundiliste vahedega;
3) iga klientrakendus käivitas skripti, mis logis kliendi automaatselt tsooni serverisse ja liigutas kliendi avatari maailmas ringi;
4) 10 minutit pärast viimase kliendi käivitamist pandi kõik kliendid ja serverid kinni ning koguti logid.

Eksperimendi tulemused

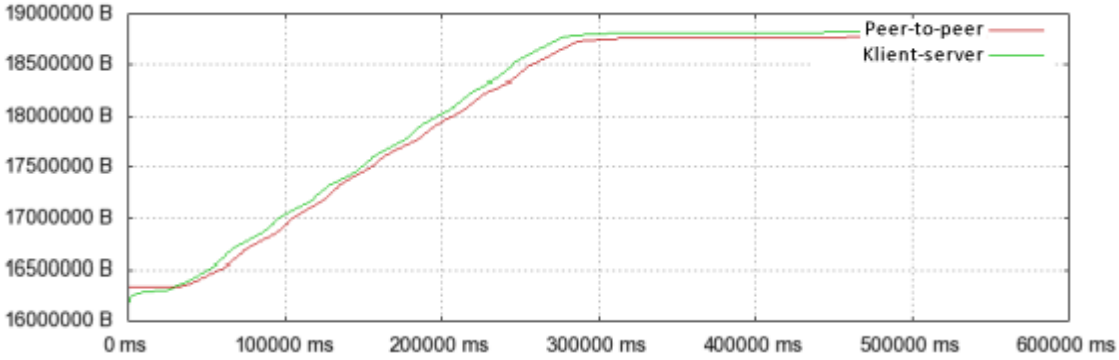
Kogutud andmete põhjal genereeriti kolm diagrammi *peer-to-peer* ja klient-server andmetega – protsessori kasutus (joonis 1), mälukasutus (joonis 2) ja akumuleeritud võrgukasutus (joonis 3). Saadud tulemuste põhjal võib öelda, et protsessori kasutus ei erine võrreldavate metoodikate puhul kuigipalju ning mõlemal juhul jääb protsessori kasutus kokkuvõttes umbes 10% juurde. Sarnaselt protsessori kasutusele ei erine ka mälukasutus suuresti *peer-to-peer* ja klient-server režiimis.

Esimesed arvestavad erinevused tulevad võrgukasutuse juures – siin on selgesti näha, et *peer-to-peer* režiimis on virtuaalse tsooni serveri võrgukasutus oluliselt väiksem kui klient-server režiimis.

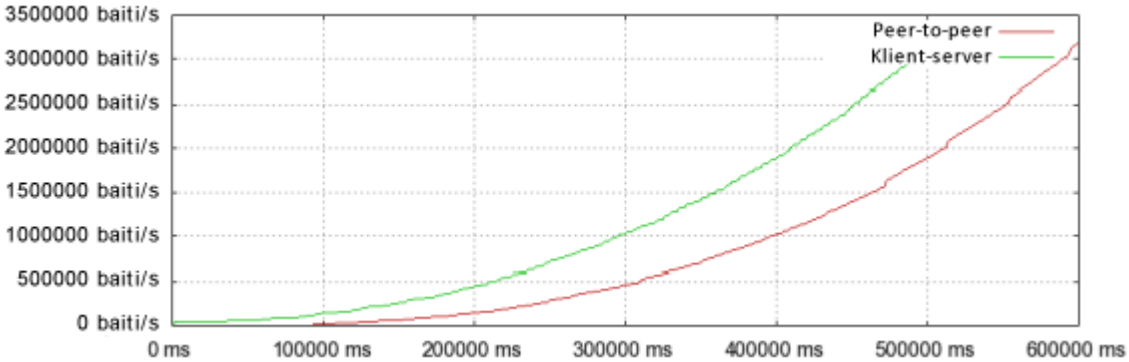
Antud eksperiment näitas, et platvormi haldamise kogukulud *peer-to-peer* režiimis ei ole suuremad kui klient-server režiimis. Kui teenusepakkuja kulud andmesidele on olulised, võib *peer-to-peer* tehnoloogia aidata kulusid kokku hoida.



Joonis 1. Protsessori kasutus peer-to-peer ja klient-server režiimis.



Joonis 2. Mälu kasutus peer-to-peer ja klient-server režiimis.



Joonis 3. Akumuleeritud võrgukasutus peer-to-peer ja klient-server režiimis.

4.3.2 Eksperiment 2: Kasutajate skriptide koormusjaotuse mõju hajussimulatsioonile

Hüpotees

Kui mõni VirtualLife võrgu tipp on skriptidega üle koormatud, siis vajalik hulk skripte saab täitmiseks migreerida teistele tippudele.

Eksperimendi kirjeldus

Eksperimendi läbiviimiseks kasutati kolme erinevat riistvara konfiguratsiooni:

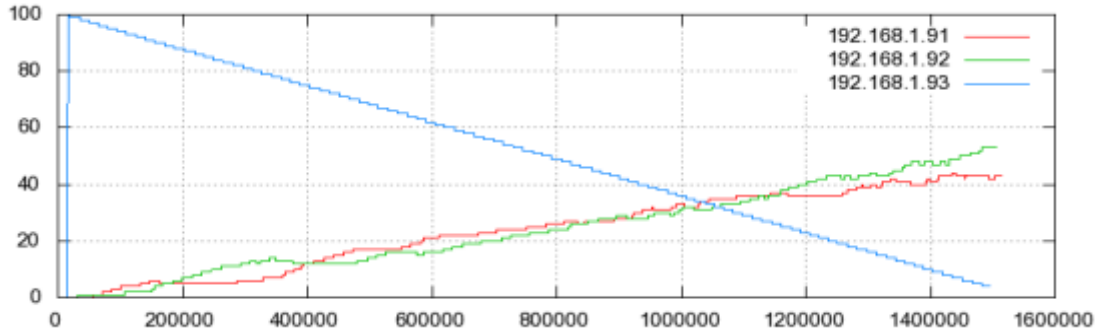
- 2 lauaarvutit, mis jooksutasid klientrakendust ning üks arvutitest ka virtuaalse riigi ja tsooni serverit;
- 3 lauaarvutit, mis jooksutasid klientrakendust ning üks arvutitest ka virtuaalse riigi ja tsooni serverit;
- kahest füüsilisest serverist koosnev klasterarvuti 16 virtuaalmasinaga, kus 15 virtuaalmasinat jooksutasid klientrakendusi ning üks virtuaalmasin jooksutas virtuaalse riigi ja tsooni serverit.

Seda eksperimenti korrati kolm korda, muutes igal korral ülekoormuse avastamise algoritmi. Eksperimendi alguses käivitas üks klientrakendustest (peremeestipp) suure hulga instantse mõõduka keerukusega skriptist ning hakkas seejärel skripti instantse teistele võrgutippudele migreerima.

Eksperimendi tulemused

Eksperiment näitas eelkõige seda, et VirtualLife'is implementeeritud koormusjaotus-mehhanismid töötasid korrektselt ning et skriptimismootor on piisavalt paindlik paljude skriptide käitamiseks.

Joonisel 4 on näha kolmel lauaarvutil tehtud eksperimendi migreerimise tulemused, kus igal võrgutipul mõõdeti skriptide arvu. Vertikaaltelg tähistab skriptide arvu ja horisontaaltelg aega millisekundites. Sinise joonega tähistatud võrgutipp (mille riistvara konfiguratsioon oli ühtlasi kõige nõrgem) migreeris peaaegu kõik temal käivitatud 100-st skriptist teistele tippudele.



Joonis 4. Skriptide arv kõikidel võrgutippudel.

4.3.3 Eksperiment 3: Virtuaalse maailma konsistentsus objektide loomise mõttes

Hüpotees

Ühes võrgutipus loodud objektid VirtualLife'i maailmas propageeruvad *peer-to-peer* režiimis teistele võrgutippudele vähemalt sama kiiresti kui klient-server režiimis.

Eksperimendi kirjeldus

Eksperimendi läbiviimiseks kasutati ühes klientrakenduses skripti, mis lõi kindla ajavahemiku jooksul objekte. Objektide tekkimise aeg logiti virtualiseeritud keskkonnas 100 millisekundi täpsusega ning korratud katsete puhul 5 millisekundi täpsusega kõigis võrgutippudes.

Eksperimendi tulemused – virtualiseeritud keskkond

Virtualiseeritud keskkonnas läbiviidud katsete tulemuste analüüsiks leiti iga objekti loomise aja standardhälbed. Väiksemad standardhälbed näitavad seda, et loodud objekt ilmus kõikidele võrgutippudele suhteliselt samaaegselt, suuremad hälbed vastupidi – objektid ilmusid võrgutippudele viivitusega.

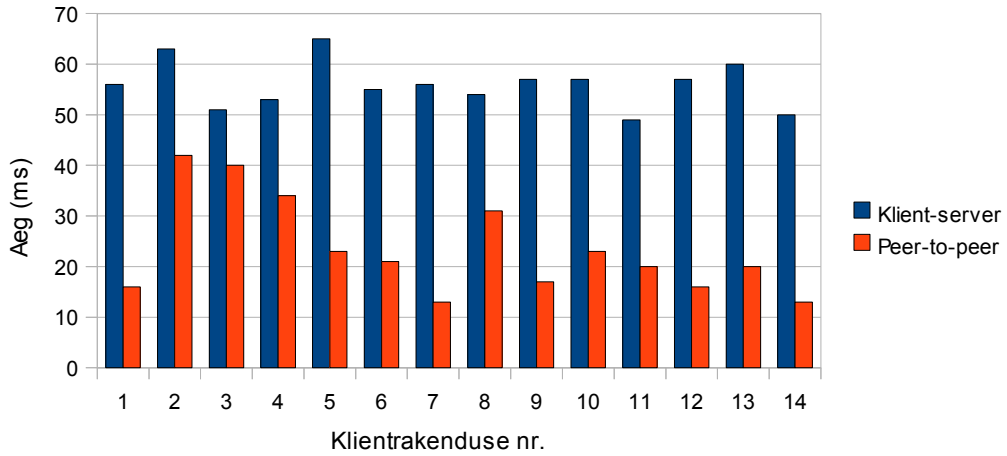
Uued objektid propageerusid *peer-to-peer* režiimis umbes sama aja jooksul kui klient-server režiimis, kuid kuna virtualiseeritud keskkond võis tulemusi mõjutada ning logimise täpsus osutus ebapiisavaks, siis ei saa siit lõplikke järeldusi teha.

Eksperimendi tulemused – korduskatse

Korduskatse puhul leiti andmete analüüsiks p*eer-to-peer* ja klient-server režiimis loodud

objekti tekkimise aeg igas klientrakenduses ning arvutati keskmine objekti tekkimise aeg mõlemal juhul.

Korduskatsel saadud tulemused olid oluliselt paremad kui virtualiseeritud keskkonnnas läbiviidud katsel. Joonisel 5 on kujutatud loodud objekti propageerumise ajad klient-rakenduste lõikes. Suured ajalised erinevused klientrakenduste vahel mõlemas režiimis on tingitud objekti loomise protsessist, mis hõlmab endas näiteks andmete lugemist kettalt. Andmete analüüsist selgus, et loodud objektid propageerivad *peer-to-peer* režiimis keskmiselt 23,5 millisekundi jooksul, seevastu klient-server režiimis 55,93 millisekundi jooksul. See tulemus näitab, et *peer-to-peer* režiimis propageerivad objektid keskmiselt 2,4 korda kiiremini.



Joonis 5. Objektide propageerimise keskmine aeg klientrakenduste lõikes.

4.3.4 Eksperiment 4: Virtuaalse maailma konsistentsus objektide asukohtade värskendamise suhtes

Hüpotees

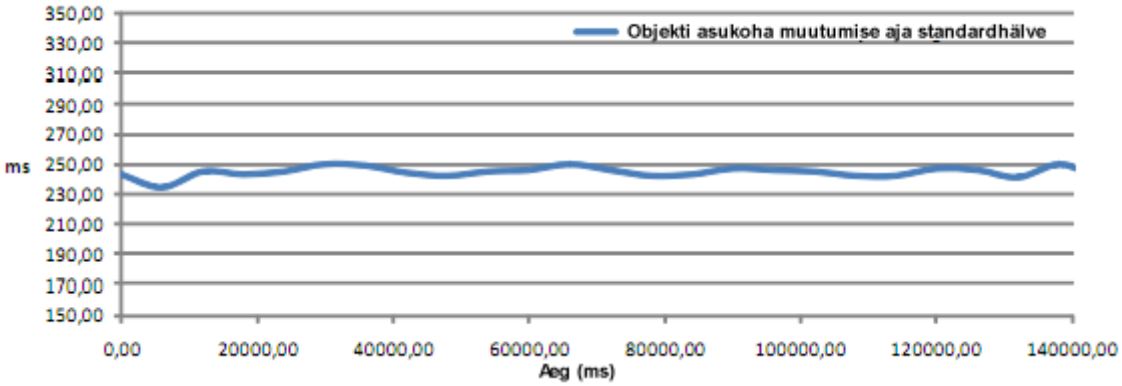
VirtuaLife maailmas ühes võrgutipus objektile tehtud muudatused propageerivad teistele võrgutippudele *peer-to-peer* režiimis vähemalt sama kiiresti kui klient-server režiimis.

Eksperimendi kirjeldus

Eksperimendi läbiviimiseks kasutati ühes klientrakenduses skripti, mis lõi ühe objekti ning hakkas seda liigutama mööda eeldefineeritud trajektoori. Eksperimendi tarvis logiti loodud objekti koordinaadid 5 millisekundilise intervalliga.

Eksperimendi tulemused – virtualiseeritud keskkond

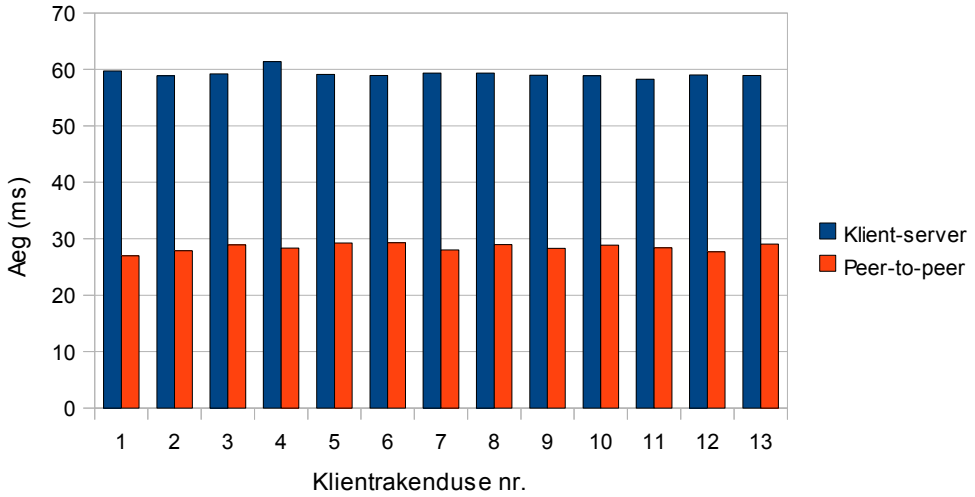
Tulemuste analüüsiks leiti iga objekti asukoha muutumise aja standardhälbed. Kui mingi objekt muudab oma asukohta ühes võrgutipus, siis vähemalt 68% teistest võrgutippudest näevad seda asukoha muutust 250 millisekundi jooksul. Lisaks on joonisel 6 näha, et objektide asukohtade uuenduste aeg on suhteliselt konsistentne terve eksperimendi jooksul.



Joonis 6. Objekti asukoha uuenduste aja standardhälve.

Eksperimendi tulemused – korduskatse

Korduskatse tulemused olid ka siin oluliselt paremad kui virtualiseeritud keskkonnas läbiviidud katsel. Tulemuste analüüsiks leiti iga asukoha muutumise aeg igas kliendis, arvutati muutumise aja keskmine väärtus ning seejärel nende keskmiste keskmine väärtus. Joonisel 7 on näha, et uuendused ilmusid *peer-to-peer* režiimis keskmiselt 2 korda kiiremini kui klient-server režiimis.



Joonis 7. Objekti asukohamuutuste keskmine propageerumise aeg peer-to-peer ja klient-server režiimis.

4.3.5 Eksperiment 5: Võrgu ribalaiuse varieerumise mõju konsistentsusele objektide asukohtade värskendamise suhtes

Hüpotees

Objektile tehtud muudatused ühes võrgutipus (peremeestipus) propageeruvad teistele tippudele vähemalt sama kiiresti kui uue objekti loomine *peer-to-peer* režiimis.

Eksperimendi kirjeldus

Selleks, et emuleerida realistlikku võrguühenduse stsenaariumit VirtualLife võrgutippude vahel, piirati võrguühenduse alla- ja üleslaadimise kiiruseid. Ribalaiused varieerusid tippude vahel 128 Kbit/s kuni 2 Mbit/s. Peremeestipp kasutas 2 Mbit/s kiirusega võrguühendust, virtuaalse riigi ja tsooni serverid kasutasid piiramata kiirusega võrguühendusi.

Eksperiment viidi läbi järgmiselt:

- virtuaalse riigi ja tsooni serverid käivitati ja tsooni server ühendus riigi serverisse;
- kõik klientrakendused logisid sisse virtuaalse tsooni serverisse;
- peremeestipus loodi objekt ning hakati seda liigutama mööda eeldefineeritud trajektoori.

Eksperimendi tulemused

Sarnaselt neljanda eksperimendi originaalkatsega arvutati välja objektide asukohtade muutumise aja standardhälbed ning antud võrguühenduse stsenaariumi puhul olid saadud ajad umbes 17% suuremad. Samal ajal näitas eksperiment, et VirtualLife platvorm saab antud tingimustes hästi hakkama, arvestades, et ühenduse ribalaius erines kuni 16 korda.

4.3.6 Eksperiment 6: Võrgu latentsuse varieerumise mõju konsistentsusele objektide asukohtade värskendamise suhtes

Hüpotees

Objektile tehtud muudatused ühes võrgutipus (peremeestipus) propageeruvad teistele tippudele vähemasti sama kiiresti kui uue objekti loomine *peer-to-peer* režiimis. Kasutusmugavus halveneb vähe.

Eksperimendi kirjeldus

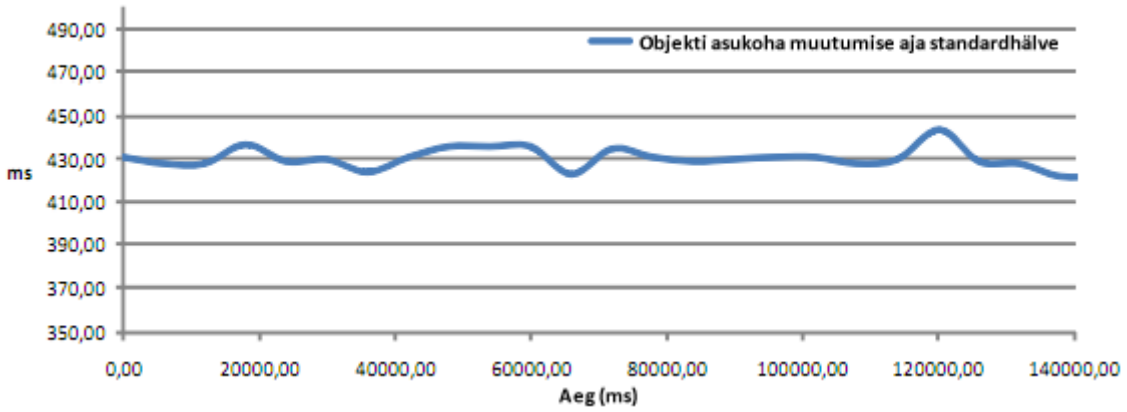
Umbes pooltele võrgutippudele lisati realistlikuma võrguühenduse emuleerimiseks 200 ms võrgulatentsi. Peremeestipp ja serverid kasutasid ilma latentsita võrgukonfiguratsioone.

Eksperiment viidi läbi järgmiselt:

- virtuaalse riigi ja tsooni serverid käivitati ja tsooni server ühendus riigi serverisse;
- kõik klientrakendused logisid sisse virtuaalse tsooni serverisse;
- peremeestipus loodi objekt ning hakati seda liigutama mööda eeldefineeritud trajektoori.

Eksperimendi tulemused

See eksperiment oli sarnane eksperimentide #4 ja #5 originaalkatsetega, kuid siin oli lisatud võrgulatentsi 200 ms. Jooniselt 8 on näha, et *peer-to-peer* režiimis on tulemused kehvemad kui eksperimendis #4 ning umbes 100 ms kehvemad kui eksperimendis #5. Samal ajal näitavad antud eksperimendi tulemused, et objektide asukohtade muutuste propageerumine on küllaltki ühtlane nii koht- kui ka laivõrgus.



Joonis 8. Objektide asukohtade muutuste propageerumise aja standardhälbed.

4.3.7 Eksperiment 7: Kasutajate skriptide koormusjaotus heterogeense võimsusega arvutivõrgus

Hüpotees

Kasutades protsessori jõudlust arvestavat võrdlusfunktsiooni, migreeruvad skriptid võimsamate protsessoritega võrgutippudele.

Eksperimendi kirjeldus

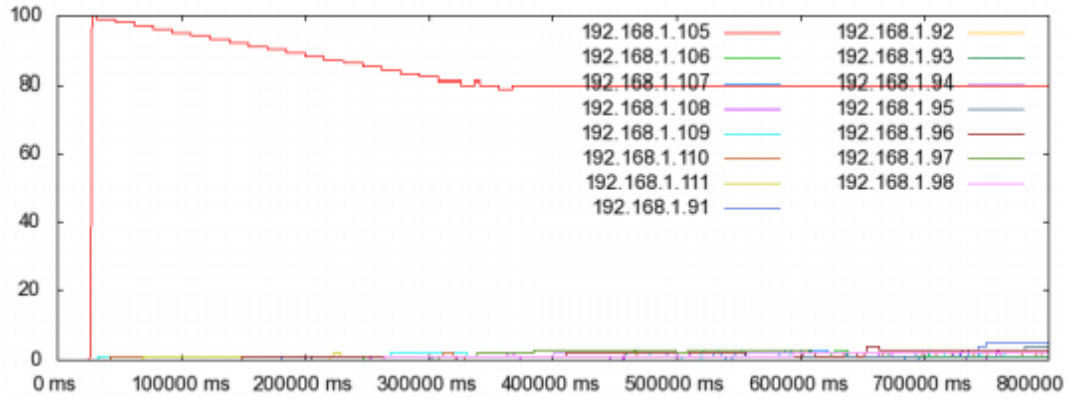
Võrgutippudel 1 kuni 8 skaleeriti protsessori jõudlus täies mahus, st. need tipud kasutasid kogu olemasolevat protsessorivõimsust. Tippudel 9 kuni 15 skaleeriti protsessori jõudlus pooles mahus, st. need tipud kasutasid poolt kogu olemasolevast protsessori jõudlusest.

Üks võrgutippudest käivitas 100 instantsi keskmist koormust genereerivast skriptist ning kasutas koormuse jaotamise mehhanismi, mis tegi otsuseid protsessi koormatuse kohta jõudlusfunktsiooni põhjal.

Eksperimendi tulemused

Kokkuvõttes migreerus skripte vähe – kõigest 20. Selline tulemus on osaliselt tingitud skriptimootorist, mis on piisavalt võimas ja paindlik, et teha ülekoormuse avastamine keeruliseks.

Joonisel 9 on näha skriptide arv igal võrgutipul eksperimendi vältel. Andmete põhjal tõdeme, et kuigi suurema võimsusega võrgutippudele migreeriti rohkem skripte, oli vahe väiksema võimsusega võrgutippude vahel liiga väike, et teha kokkuvõtlikke järeldusi.



Joonis 9. Skriptide hulk kõikidel võrgutippudel.

Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks on uurida *peer-to-peer* tehnoloogia rakendatavust virtuaalmaailmade simuleerimisel ning selgitada välja, milliste aspektide poolest sobib *peer-to-peer* arhitektuur paremini virtuaalmaailmade simuleerimiseks kui klient-server arhitektuur. Lisaks uuriti, kuidas mõjutab nimetatud tehnoloogiate rakendamine teenusepakkuja kogukulusid virtuaalmaailmade haldamisel.

Töö raames korraldati eksperimendid teadusuuringute tarbeks loodud virtuaalmaailma keskkonnaga VirtualLife, mida on võimalik seada tööle nii klient-server kui ka *peer-to-peer* režiimis. Eksperimendid viidi läbi virtualiseeritud keskkonnas, kuid töö käigus korrati kahte olulisemat katset ka füüsilisestest arvutitest koosnevas keskkonnas.

Eksperimendid 1 ja 2 näitasid, et platvormi haldamise kogukulud *peer-to-peer* režiimis ei ole suuremad kui klient-server režiimis ning et süsteemi koormusjaotusmehhanismid töötasid korrektselt.

Eksperimendid 3 ja 4, mis viidi läbi nii virtualiseeritud keskkonnas kui ka füüsilistest arvutitest koosnevas keskkonnas, näitasid, et *peer-to-peer* režiimis propageeruvad maailmas tehtud uuendused vähemalt sama kiiresti kui klient-server režiimis. Veel enam, eksperimentide kordamisest füüsilistest arvutitest koosnevas keskkonnas selgus, et muudatused propageeruvad *peer-to-peer* režiimis isegi umbes kaks korda kiiremini.

Eksperimendid 5 ja 6 näitasid, et VirtualLife platvorm saab mõlemas režiimis võrdselt hästi hakkama ka erineva võrgukiiruse ja latentsiga klientide puhul. Eksperiment 7 näitas, et platvormi arhitektuur ja koormusjaotuse mehhanismid võimaldavad kasutajatel luua suurel hulgal skripte, hoides samal ajal kasutajakogemuse konsistentsena.

Tuginedes eksperimentide tulemuste analüüsile võime väita, et *peer-to-peer* tehnoloogia rakendamine virtuaalmaailmades võimaldab pakkuda paremat kasutajakogemust ning aitab vähendada teenusepakkuja kogukulusid, samas potentsiaalselt suurendades kasutajate kulusid internetile ja arvutusressursile.

Summary

Bachelor's thesis

Siim Annuk

Applying Peer-to-peer Technology In Virtual Worlds

The purpose of the given thesis is to investigate the usability and effectiveness of peer-to-peer architecture in contrast to traditional client-server architeture when simulating virtual worlds. In addition, we study how peer-to-peer helps to reduce the cost of ownership for the service provider as opposed to the traditional client-server architecture.

A number of experiments were conducted with scientific research project VirtualLife using both peer-to-peer and client-server architectures. The experiments were conducted on a special-purpose cluster computer consisting of multiple virtual nodes. Some experiments were also repeated on a non virtualized environment consisting of physical computers, since it was found that the virtualized environment might have had a negative effect on the performance of the simulation.

Based on the experiment results we conclude that using peer-to-peer architecture when simulating virtual worlds helps to provide a better user experience while reducing the cost of ownership for the service provider. We also confirm that the usage of peer-to-peer architecture might increase the bandwidth load and resource usage for the client participating in the virtual world.

Kasutatud kirjandus

[1] F. Biocca, M. Levy. Communication in the Age of Virtual Reality. 1995.

[2] I. Sutherland. “The Ultimate Display”, Proceedings of IFIP Congress. 1965.

[3] DigiBarn Historic Games Events: Maze on Imlac PDS-1D at Nov 6, 2004 event. http://www.digibarn.com/collections/games/maze-war/imlacs-pds1-maze/index.html, [07.05.2012].

[4] D. Frey, J. Royan, R. Piegay, A. M. Kermarrec, E. Anceaume, F. Le Fessant. Solipsis: A Decentralized Architecture for Virtual Environments. Proceedings of International. Workshop on Massively Multiuser Virtual Environments (MMVE), pp. 29–33, 2008.

[5] Olivier Beaumont, Anne-Marie Kermarrec, and Etienne Rivière. Peer to peer multidimensional overlays: Approximating complex structures. Technical report. INRIA.

[6] K. Graffi, C. Gross, D. Stingl, D. Hartung, A. Kovacevic, R. Steinmetz. LifeSocial.KOM: A Secure and P2P-based Solution for Online Social Networks. IEEE CCNC’11.

[7] Ashwin Bharambe, John R. Douceur, Jacob R. Lorch, Thomas Moscibroda, Jeffrey Pang, Srinivasan Seshan, and Xinyu Zhuang. Donnybrook: Enabling large-scale, high-speed, peer-to-peer games. Proceedings of the Conference on Computer Communications (SIGCOMM). Association for Computing Machinery, Inc. 2008

[8] Shun-Yun Hu, Chuan Wu, Eliya Buyukkaya, Chien-Hao Chien, Tzu-Hao Lin, Maha Abdallah and Jehn-Ruey Jiang. "VAST: A Spatial Publish Subscribe Overlay for Massively Multiuser Virtual Environments," VAST Technical Report (VAST-TR-03-2010), Mar. 2010.

[9] Ilja Livenson. VirtualLife Security Infrastructure. Magistritöö. Tartu Ülikool, Matemaatikainformaatikateaduskond, Arvutiteaduse instituut. 2009.

[10] Dan Bogdanov, Ilja Livenson. VirtualLife: Secure Identity Management in Peer-to-Peer Systems. Proceedings of the 1st International ICST Conference on User Centric Media (UCMedia2009), LNICST vol 40, Springer, 2009.

[11] Siim Annuk, Dan Bogdanov, Kevin Glass, Roman Jagomägis, Ilja Livenson, Jaak Ristioja, Gian Marco Todesco. D12.3 Peer-to-peer validation report. June 2011.