VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Duomenų srauto optimizacijos internetiniams žaidimams

Data Traffic Optimizations for Online Games

Magistro baigiamasis darbas

Atliko: Dainius Kreivys (parašas)

Darbo vadovas: lekt. Irmantas Radavičius (parašas)

Recenzentas: doc. Sigitas Dapkūnas (parašas)

Vilnius – 2017

Kreivys D. Duomenų srauto optimizacijos internetiniams žaidimams: programų sistemų magistro darbas / vadovas lekt. I. Radavičius; Vilniaus Universitetas, Matematikos ir informatikos fakultetas, Programų sistemų katedra. – Vilnius, 2017. – 65 p.

Duomenų srauto optimizacijos internetiniams žaidimams

Santrauka

Žaidėjų skaičius kasmet vis auga, o aptarnauti tūkstančius ar šimtus tūkstančių žaidėjų internetiniame žaidime reikalauja tam tikrų techninių sprendimų, į ką įeina duomenų srauto (angl. *bandwidth*) kontroliavimas bei optimizavimas.

Atliktame darbe aiškinamasi kokiu būdu reikia perdavinėti duomenis, kaip juos suspausti prieš perdavimą ir kaip rečiau juos perduoti. Ieškoma sprendimų kombinacija, kuri padėtų mažinti duomenų srautą internetiniams žaidimams. Aprašomi gauti bandymų rezultatai ir rekomendacijos kaip pritaikyti sprendimus internetiniams žaidimams.

Raktiniai žodžiai

Duomenų srauto optimizacijos, internetiniai žaidimai, MMORPG, TCP, ROHC, SCTP, Huffman, LZ4, SIMD, FastPFOR.

Kreivys D. Data Traffic Optimizations for Online Games: Master's work in Software Engineering / supervisor lect. I. Radavičius; Vilnius University, Faculty of Mathematics and Informatics, Departament of Software Engineering. – Vilnius, 2017. – 65 p.

Data Traffic Optimizations for Online Games

Summary

Number of players continues to grow each year, and to serve thousands or hundreds of thousands of players in an online game requires some technical solutions like data traffic control and optimization.

The work examines how to compress, transfer and send less data over the network. In this work author is searching for a combination of solutions that will help reduce the data traffic for online games. The results of the investigation are described and recommendations on how to use the solutions for online games are proposed.

Keywords

Data traffic optimization, online games, MMORPG, TCP, ROHC, SCTP, Huffman, LZ4, SIMD, FastPFOR.

TURINYS

LENTELIŲ IR PAVEIKSLŲ SĄRAŠAS

Lentelės:

Paveikslai:

ĮVADAS

Kompiuteriniai žaidimai - tai visavertė šiuolaikinės kultūros dalis. Pasaulinė kompiuterinių žaidimų pramonė kiekvienais metais sparčiai plečiasi, ši sritis vis giliau įsiterpia į kasdienį mūsų gyvenimą taip pat, kaip ir literatūra ar kinas. Kaip paminėta straipsnyje [Npd13], dabar dauguma žaidėjų žaidžia internete ir žaidėjų skaičius didėja kiekvienais metais.

Daugelio žaidėjų žaidimas internete (angl. *Massive Multiplayer Online Game, MMOG*) yra toks internetinis žaidimas, kuris yra pajėgus aptarnauti šimtus ar tūkstančius žaidėjų vienu metu. Šiems žaidimams yra būtinas internetas. Daugelis žaidimų turi bent vieną pastovų virtualų pasaulį, tačiau kiti žaidimai tiesiog turi daug žaidėjų, konkuruojančių tarpusavyje vienokia ar kitokia forma.

MMO žaidimai suteikia galimybę žaidėjams bendradarbiauti ir konkuruoti vieniems su kitais virtualiame pasaulyje, o kartais ir prasmingai bendrauti su žmonėmis visame pasaulyje. Internetiniai žaidimai leidžia žaidėjams vienu metu prisijungusiems prie žaidimo serverio bendrauti ne tik tarpusavyje, bet ir su žaidimo aplinka, kuri gali susidėti iš statinių bei dinamiškai kintančių duomenų, kuriuos taip pat reikia atnaujinti klientų programose, kas papildomai didina duomenų srautą.

Žaidėjų skaičius kasmet vis auga, o aptarnauti tokį didelį, 1 lentelėje matomą, žaidėjų skaičių internetiniame žaidime reikalauja tam tikrų techninių sprendimų [SS13], į ką įeina duomenų srauto (angl. *bandwidth*) ir gaišties laiko (angl. *latency*) kontroliavimas bei optimizavimas.

1 lentelė. Dabartiniai, populiariausi internetiniai žaidimai pagal žaidėjų skaičių (šaltinis: http://steamcharts.com/)

Žaidimo pavadinimas	**Duomenų paėmimo metu prisijungusių žaidėjų skaičius**	**Didžiausias vienu metu prisijungusių žaidėjų skaičius**
Dota 2	676 987	965 737
Counter-Strike: Global	407 460	823 694
Team Fortress 2	57 493	81 671
Sid Meier's Civilization V	30 380	58 848
Garry's Mod	26 881	60 907

Didėjant žaidėjų skaičiui taip pat didėja ir perduodamų duomenų srautas, kas atsiliepia fizinio serverio resursų naudojimui, taipogi tai didina žaidimo tinklo gaišties laiką, kas neigiamai paveikia

žaidimo funkcionavimą. Vis didėjantis duomenų srautas yra esama problema [PWA11], todėl turi būti imtasi naujų priemonių, kad būtų išlaikytas stabilus žaidimo funkcionalumas.

Analitinėje dalyje aiškinama kaip optimizuoti duomenų srautą internetiniams žaidimams. Tam atlikti yra keliami trys klausimai:

1) Kokiu būdu reikia perdavinėti duomenis?
2) Kaip suspausti perduodamus duomenis?
3) Kaip rečiau perduoti duomenis?

Pirmam klausimui atsakyti nagrinėjami transporto protokolai. Internetinių žaidimų tinklo protokolai nustato kada, kaip ir kokiu formatu žaidimo būsenos yra atnaujinamos serveryje bei visuose tuo metu prisijungusiuose klientuose. Tai yra internetinio žaidimo dalis, kuri turi būti kruopščiai suplanuota ir realizuota, kad vėliau būtų apsisaugoma nuo rizikų dėl papildomo darbo žaidimo serveriui ir klientui, kas gali negatyviai paveikti pasitenkinimą žaidimu per internetinius delsos (angl. *lag*) ir defektų (angl. *bug*) atsiradimus.

Paketinio duomenų perdavimo tinkle (angl. *packet-switched network*) siunčiamus duomenis vadiname paketais, kurie tinkle suskaidomi į segmentus. Kiekvienas segmentas susideda iš antraštės ir dalies perduodamų duomenų. Skirtingi transporto protokolai turi skirtingus savo antraščių dydžius.

Internetiniams žaidimams yra svarbios šios paketų savybės: dydis, sparta ir siuntimo dažnis. Paketo dydis susideda iš antraštės ir perduodamų duomenų dydžių. Jis įtakoja paketo perdavimo spartą, kuo paketas didesnis, tuo lėčiau jį gaus klientas, taip didėja žaidimo gaišties laikas ir dėl to atsiranda žaidimo jautrumo uždelsimas (angl. *delayed sensitivity*). Didelis paketų perdavimo dažnis, apie kurį bus kalbama šiame tekste vėliau, įtakoja duomenų srautą, kuris paveikia žaidimo gaišties laiką. Šios savybės yra labai svarbios žaidimo funkcionalumui ir pasitenkinimui.

Iškeliami kriterijai, kad tam tikri transporto protokolų trūkumai, tokie kaip duomenų paketų perdavimo užtikrinimo, paketų eiliškumo nebuvimas ar per didelis paketo antraštės dydis netrukdytų kokybiškai aptarnauti žaidėjus.

Aiškinami kiekvienų pasirinktų architektūrų skirtumai, jų privalumai bei trūkumai, ir yra siekiama įsivertinti pasirinktų technologijų realizacija internetiniams žaidimams. Taip pat yra siekiama surasti tinkamo būdo, kuriuo būtų išlaikomas optimalus duomenų srautas perdavinėjant duomenys tarp žaidimo klientų ir serverio.

Antram klausimui, kaip suspausti perduodamus duomenis, atsakyti nagrinėjamos duomenų suspaudimo technologijos. Paketo turinio dydžiui valdyti reikalingas suspaudimo algoritmas, kurio efektyvumas būtų vertinamas pagal suspaudimo laipsnio, skaičiavimo laiko, į kurį įeina duomenų

suspaudimas ir atspaudimas, bei suspausto paketo perdavimo laiko santykiu. Jei gautas santykis yra mažesnis už nesuspausto paketo dydžio ir perdavimo laiko santykį, tuomet suspaudimo algoritmas yra efektyvus. Mažinant paketo dydį bus didinama kita svarbi efektyvaus paketo perdavimo savybė – sparta. Kuo mažesnis paketas, tuo greičiau jis yra perduodamas.

Keliamas kriterijus, kad suspaudimo skaičiavimo laikas nesukeltu didelės delsos internetiniams žaidimams, o duomenų paketų antraščių suspaudimai neigiamai neįtakotų paketų perdavimo užtikrinimą.

Aiškinamos technologijos, kurios padėtų suspausti perduodamus duomenis kartu su duomenų paketų antraštėmis. Tokiu būdu siekiama išgauti optimalų perduodamų duomenų dydį.

Trečiam klausimui, kaip rečiau perduoti duomenis, atsakyti nagrinėjami metodai, kurie padeda mažinti duomenų paketų perdavimo dažnį (angl. *packet rate*) [SS13]. Šis dažnis yra svarbi duomenų perdavimo savybė, kuri randama apskaičiavus siunčiamų iš serverio į klientus ir atgal gaunamų paketų skaičių per tam tikrą laiką. Nuo jo priklauso duomenų srautas – kuo didesnis paketų siuntimo dažnis tuo daugiau bus naudojama duomenų srauto. Paketų perdavimo dažnį įtakoja skirtingi žaidimo tipų elementai, tokie kaip: žaidėjo duomenų, žaidimo aplinkos užkrovimas, kautynės, žaidėjų pozicijos atnaujinimai ir kiti.

Aiškinama kaip tinkamai realizuoti metodus, kurie mažina paketų perdavimo dažnį internetiniams žaidimams, nesukeliant negatyvių faktorių žaidimui, smarkiai neapribojant žaidėjų ir nesumažinant pasitenkinimo žaidimu.

Šio darbo **tikslas** yra sukurti sprendimų kombinaciją, skirtą optimizuoti duomenų srautui internetiniams žaidimams.

Projektinėje dalyje yra realizuojami sprendimai, kuriuos ištyrus būtų galima vertinti duomenų srauto optimizacijos laipsnį ir naudą internetiniams žaidimams.

Tiriama, kokia sprendimų kombinaciją būtų efektyviausia iš darbe tiriamų technologijų bei metodų, optimizuojant duomenų srautą internetiniams žaidimams.

Darbo uždaviniai:

1. Realizuoti duomenų paketų perdavimą skirtingais transporto protokolais, kuriais pasinaudojus bus galima atlikti aktualius temai transporto protokolų tyrimus;
2. Realizuoti duomenų paketų antraščių suspaudimo technologijas, kuriomis pasinaudojant bus galima atlikti duomenų paketų antraščių suspaudimo tyrimus;

3. Realizuoti duomenų paketų apkrovos suspaudimo technologijas, kuriomis pasinaudojant bus galima atlikti duomenų paketų apkrovos suspaudimo tyrimus;
4. Atlikti technologijų tyrimus, kurie galėtų pateikti reikiamus rezultatus;
5. Surinkti tyrimų rezultatus, iš kurių būtų galima vertinti technologijos naudą duomenų srauto optimizacijai;
6. Atlikti rezultatų analizę, taip surandant tinkamiausią sprendimų kombinaciją, kuri būtų skirta optimizuoti duomenų srautą internetiniams žaidimams;
7. Pateikti išvadas, iš kurių būtų galima aiškiai matyti ar sprendimai teikia naudą.

1. Transporto protokolų analizė

Internetiniai žaidimai naudoja skirtingos rūšies informaciją, perduodant duomenis tarp žaidimo serverio ir kliento – vieni internetiniai žaidimai perduoda įvykių žinutes, kas konkrečiu momentu nutiko pas tam tikrą žaidėją, o kiti perduoda visus pasikeitimus virtualioje žaidimo aplinkoje.

Perduodant žaidimo įvykius yra svarbu, kad visi duomenų paketai pasieks gavėją, tam, kad žaidimo būsena (realiu laiku visi vykstantys veiksmai žaidime) tarp serverio ir kliento nesiskirtų. Taip pat šiam duomenų perdavimo būdui yra svarbu, kad visi duomenų paketai pasieks savo gavėjo tokia pat eilės tvarka, kuria buvo išsiųsti, nes kitu atveju žaidimo būsena gali skirtis tarp serverio ir kliento nors ir visi duomenys būtų gauti.

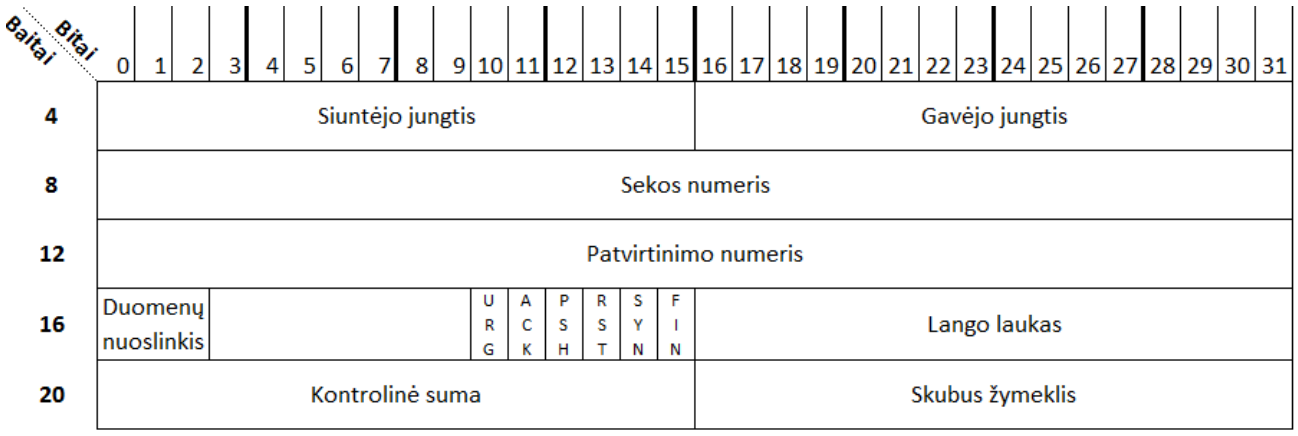
Internetinio žaidimo būsenos pasikeitimai yra perduodami žaidėjams labai dažnai ir periodiškai, nes žaidimuose, kurie naudoja tokios rūšies informaciją perdavimui, žaidimo veiksmai atliekami gan sparčiai. Tokiu atveju yra svarbu perduoti pasikeitimus kuo sparčiau, o jei vienas žaidimo būsenos pasikeitimų paketas nepasiektų gavėjo, reiktų, kad žaidimo klientas neprašytų atsiųsti jų pakartotinai, o apdorotų naujausiai gautą pasikeitimų duomenų paketą. Taip pat tokios rūšies duomenis nėra būtina priimti eilės tvarką, kad nepasidarytų grūstis kanalo gale, svarbiausia, kad būtų sparčiai perduodami duomenų paketai, o žaidimo klientas naudotų naujausiai gauto paketo duomenis.

Šie du skirtingi duomenų perdavimo būdai, tarp žaidimo serverio ir kliento, yra pasiekiami dvejais tinklo protokolais – TCP ir UDP [Bra13]. Įvykių siuntimas dažniausiai yra atliekamas per TCP, o žaidimo būsenos pasikeitimai yra perduodami naudojant UDP transporto protokolą.

1.1. TCP

TCP transporto protokolas (angl. *Transmission Control Protocol*), priešingai nei UDP, užtikrina duomenų paketų perdavimą. Priimant duomenų paketus iš siuntėjo yra išsiunčiamas atgal patvirtinimo paketas (angl. *acknowledgment packet*), kuriame yra nurodoma informacija apie gautus duomenų paketus [CHH+15]. Kai siuntėjas nesulaukia iš gavėjo atsakymo (patvirtinimo paketo), per tam tikrą laiką, duomenų paketas yra pakartotinai išsiunčiami gavėjui. TCP transporto protokolas taip pat užtikrina duomenų paketų eiliškumą, prie antraštės pridedant paketo eilės numerį, pagal kurį gavėjas žinos kuriuos duomenų paketus turi priimti pirmiau. Tokios protokolo savybės reikalauja didesnės duomenų paketo antraštės, o antraštės dydis įtakoja paketų perdavimo spartą.

TCP segmentas yra siunčiamas kaip duomenų paketų kelio schema (angl. *datagram*) [Pos81]. Interneto protokolo (IP) antraštė perduoda keletą duomenų laukų, į kuriuos įeina siuntėjo bei gavėjo IP adresai, kartu ir TCP antraštės dydis su apkrovos (angl. *payload*) dydžiu. TCP antraštė, matoma 1 pav., tinklo schemoje yra prijungiama po IP antraštės. Ji saugo specifinę informaciją skirta TCP protokolui [Pos81]: siuntėjo bei gavėjo jungtys (angl. *port*), sekos numeris (angl. *sequence number*), kuris skirtas nustatyti ar paketas gautas reikiama tvarka, patvirtinimo numeris (angl. *acknowledgment number*) skirtas perspėti siuntėją, kad sėkmingai gavo reikiamą paketą, duomenų poslinkis (angl. *data offset*) – nurodo nuo kurio bito prasideda paketo apkrova. Toliau seka rezervuotos vietos laukas (angl. *reserved*) – daug informacijos apie šį lauką nėra, dauguma straipsnių skelbia, kad šiame lauke yra rezervuoti nenaudojami 6 bitai. Po to eilės tvarka eina kontroliniai bitai (angl. *control bits*), dar kitaip vadinami žymos bitai, kurie nurodo kas yra siunčiama pakete: „URG“ žyma nurodo, kad paketas turi svarbią informaciją, „ACK“ žyma nurodo, kad siunčiamas paketas yra patvirtinimo paketas, „PSH“ žyma nurodo, kad paketas turi būti išsiunčiamas iš karto nelaukiant kitų duomenų, „RST“ žyma nurodo, kad iškilo klaida sraute ir reikalinga pradėti naują ryšio užmezgimą su gavėju, „SYN“ žyma skirta sinchronizuoti sekos numerius tarp siuntėjo ir gavėjo, „FIN“ žyma nurodo, kad baigiama komunikacija su gavėju. Po kontrolinių bitų toliau eina langos laukas (angl. *window*), kuris nurodo kiek gavėjas nuskaitys bitų, kontrolinė suma (angl. *checksum*) skirta tikrinant klaidas ar yra gauta visa reikiama informacija pateikta TCP antraštėje ir apkrovoje, skubus žymeklis (angl. *urgent pointer*) - saugo informacija, kuri nurodo nuo kurio bito prasideda ir baigiasi svarbi informacija. Toliau eina nustatymų (angl. *options*) laukas – jame gali būti pateikta informacija susijusi su maksimaliu segmento dydžiu, kiek gavėjas gali nuskaityti bitų, taip pat galima įjungti pasirinktąjį patvirtinimo mechanizmą (angl. *selective acknowledgement*) – tipiniu atveju, kai yra išsiunčiami keli duomenų paketai ir vienas iš jų nepasiekia gavėjo, tai siuntėjui gavus pranešimą, kad vis dar negavo naujo paketo, tai siuntėjas persiunčia visus iki to paketo turėtus išsiųsti paketus. Tačiau būna situacijų, kad pradingsta tik vienas paketas, o visų sekančiu nereikia persiųsti iš naujo, todėl yra galimybė įjungti šį nustatymą, kuris perspėja siuntėja, kad trūksta būtent to vieno paketo, o ne visų iki jo išsiųstų.



1 pav. TCP transporto protokolo antraštės segmento schema

Tipinė TCP antraštė užima minimaliai 20 baitų ir priklausomai nuo papildomų nustatymų gali siekti net iki 60 baitų viso duomenų paketo dydžio. Prie jos prisideda 20 baitų IP antraštės ir taip tipinė TCP/IP antraštė užima 40 baitų.

Naglio algoritmas [Nag84] buvo sukurtas perduodamų duomenų paketų skaičiaus mažinimui, kai siunčiama daug nedidelių paketų, kurių apkrova užima mažai vietos, o antraštė užima didžiąją dalį viso duomenų paketo dydžio. Paketų duomenys yra sujungiami į vieną didesnį paketą, kol yra pasiekiama tam tikra riba ar dydis ir jiems yra priskiriama viena bendra antraštė prieš siunčiant gavėjui.

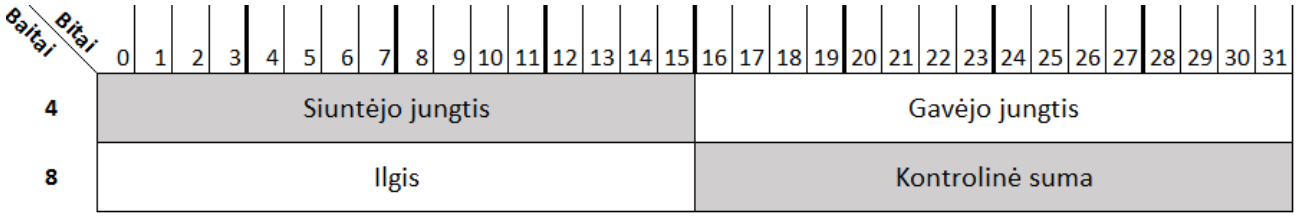
Atidėti patvirtinimo paketus (angl. *delay acknowledgment packets*) yra dar vienas metodas skirtas sumažinti perduodamų paketų skaičių. Yra daroma prielaidą, kad siuntėjui išsiuntus duomenų paketą gavėjas tuojau pat turės išsiųsti tam tikrus duomenis atgal siuntėjui. Tokiu atveju galime patvirtinimo paketą išsiųsti ne iš karto, kai gavėjas gauna duomenis, tačiau kai gavėjas paruošia duomenis siųsti atgal siuntėjui, ir tik tuomet prijungti patvirtinimo paketą prie gavėjo išsiunčiamų duomenų - taip yra sutaupomas vienos antraštės išsiuntimas.

Naudojant abi šias optimizacijas gali sukelti didelę delsą sistemos pusėje. Jeigu tik vienas nedidelis duomenų paketas bus siunčiamas, Naglio algoritmas sustabdys siuntimą 200 milisekundžių, nes turi laukti daugiau duomenų. Taip pat atidedant patvirtinimo paketus, gavėjas turės palaukti 100-200 milisekundžių kol patvirtinimo paketas išsisiųs, jei gavėjas neturės jokių perduodamų duomenų siuntėjui.

1.2. UDP

Siunčiant duomenų paketus per UDP (angl. *User Datagram Protocol*) transporto protokolą nėra sukuriama sesija tarp siuntėjo ir gavėjo [SS13]. Tinklo transporto protokolo lygyje siuntėjui nėra galimybės žinoti ar visi duomenys pasieks gavėją ir ar jie išsisiųs eilės tvarka. Jei reikia, kad būtų užtikrintas duomenų gavimas ir eiliškumas, tą įmanoma atlikti tinklo taikomųjų programų lygmenyje (angl. *application layer*) [Bra13], tačiau dažniausiai tai sulėtina duomenų perdavimo spartą.

UDP duomenų paketo antraštė, matoma 2 pav., tinklo schemoje taip pat yra prijungiama po IP antraštės. Ji saugo specifinę informaciją skirta UDP protokolui: siuntėjo bei gavėjo jungtys, antraštės ir apkrovos ilgis, sekos numeris, kuriuo galima būtų tikrinti apkrovos integralumą tinklo taikomųjų programų lygmenyje, tačiau šis laukas yra neprivalomas UDP/IP tinklo schemoje. Siuntėjo jungtis taip pat yra neprivalomas laukas, kurio atsisakius jungčiai yra priskiriama 0 reikšmė.



2 pav. UDP transporto protokolo antraštės schema

UDP antraštė užima 8 baitus viso duomenų paketo dydžio. Prie jos prisideda 20 baitų IP antraštė ir taip UDP/IP antraštė užima 28 baitus.

UDP transporto protokolas yra dažniausiai naudojamas pirmo asmens šaudyklės žaidimuose, tokiuose kaip: „Quake 3“, „Counter-Strike“, „Call of Duty“, „Half Life“. Šiuose žaidimuose nėra būtina užtikrinti ar visi perduodami duomenų paketai pasieks žaidimo klientą, nes yra naudojamas naujausiai gautas duomenų paketas, žaidimo būsenos atnaujinimui. Priešingu atveju, pradingus duomenų paketui, žaidimo būsena turėtų būti pakartotinai išsiunčiama, nors jau būtų paruoštas duomenų paketas siuntimui su naujesniais žaidimo būsenos atnaujinimais. Todėl naudojant UDP transporto protokolą, jei pasitaikytų, kad tam tikras duomenų paketas nepasiektų gavėjo, tai gavėjas naudotų naujausiai gautąjį duomenų paketą, o pasitelkus žaidimo veiksmų nuspėjimo technologijas [WKV+06] būtų atnaujinama žaidimo būsena.

Yra galimybė UDP protokolą naudoti kartu su TCP – vienu protokolu būtų perduodami žaidimo duomenis reikalaujantis mažesnės delsos tačiau nereikalaujantis pristatymo užtikrinimo, o kitu protokolu - priešingo konteksto duomenis.

1.3. TCP ir UDP protokolai internetiniuose žaidimuose

Interneto kompiuterinis žaidimas vaidmenimis, kurį žaidžia daug žaidėju vienu metu – MMORPG (angl. *Massively Multiplayer Online Role-Playing Game*), tai toks internetinių žaidimų žanras, kurio populiarumas auga kiekvienais metais. Tokie žaidimai turi virtualius pasaulius, kuriuose žaidžia tūkstančiai žaidėjų, kiekvienas jų valdo tam tikrą avatarą. Žaidėjai turi vykdyti tam tikras misijas, rinkti artifaktus. Pasiekus tam tikrą žaidimo lygi žaidėjai gali tobulinti savo sugebėjimus.

Tokie žaidimai tampa vis labiau populiaresni, o labiausiai žinomas yra „World of Warcraft“, sukurtas Blizzard Ent. Kaip straipsnis rašo yra pasiekęs 12 milijonų žaidėjų skaičių [SDM09]. Su tokiu dideliu skaičiumi žaidėju atsiranda problemų ties duomenų srautu ir pasidaro sunku aptarnauti visus žaidėjus.

Nors MMORPG žaidimai yra populiariausi iš visų internetinių žaidimų žanrų, taip pat yra ir kitų ganėtinai populiarių žanrų. Kaip pavyzdžiui pirmo asmens šaudyklės – FPS (angl. *First Person Shooter*) juos taip pat sieja bendrų žaidimo charakteristikų: virtualiame pasaulyje žaidžia iki kelių dešimčių žaidėjų, kurie naudoja šaunamuosius ginklus pasiekti tam tikrus žaidimo tikslus.

Šių žanrų skirtumai yra plačiai išnagrinėti [SKR07][OH03]. Pagrindiniai skirtumai gali būti apibendrinami:

- MMORPG žaidimų sesijų trukmė yra ilgesnė nei FPS [SDM09]. Tačiau FPS žaidėjai žaidžia kelis raundus per žaidimo sesiją.
- Žaidėjų skaičius FPS virtualiosiose pasauliuose siekia kelias dešimtis, kur MMORPG virtualius pasaulius dalinasi tūkstančiai žaidėjų.
- FPS žaidimai reikalauja realaus laiko spartumo, kas yra labai svarbu. MMORPG žaidimams, tuo tarpu spartus veiksmai nėra tokie svarbūs, nes prieš kiekvieną veiksmą žaidėjas turi planuoti savo veiksmų seką (pvz. pele pasirenkamas objektas iš sąrašo ir pasirenkamas norimas veiksmas ką su tuo objektu daryti).
- FPS žaidimai naudoja UDP protokolą, o daugiausiai MMORPG žaidimų naudoja TCP.

Paskutinis svarbiausias skirtumas yra tai, kad UDP protokolas FPS žaidimuose neperduoda pakartotinai duomenų kai duomenų paketas yra prarandamas. Šis funkcionalumas turi didelę pasekmę žaidimuose, pvz. šaunamo ginklo kulka gali būti prarasta ir nenukeliauti klientui. Kai kurie žaidimai naudoja užslėpimo algoritmus, kad nuslėptų nesutapimus tarp žaidėjų [ZA04]. Spartus duomenų

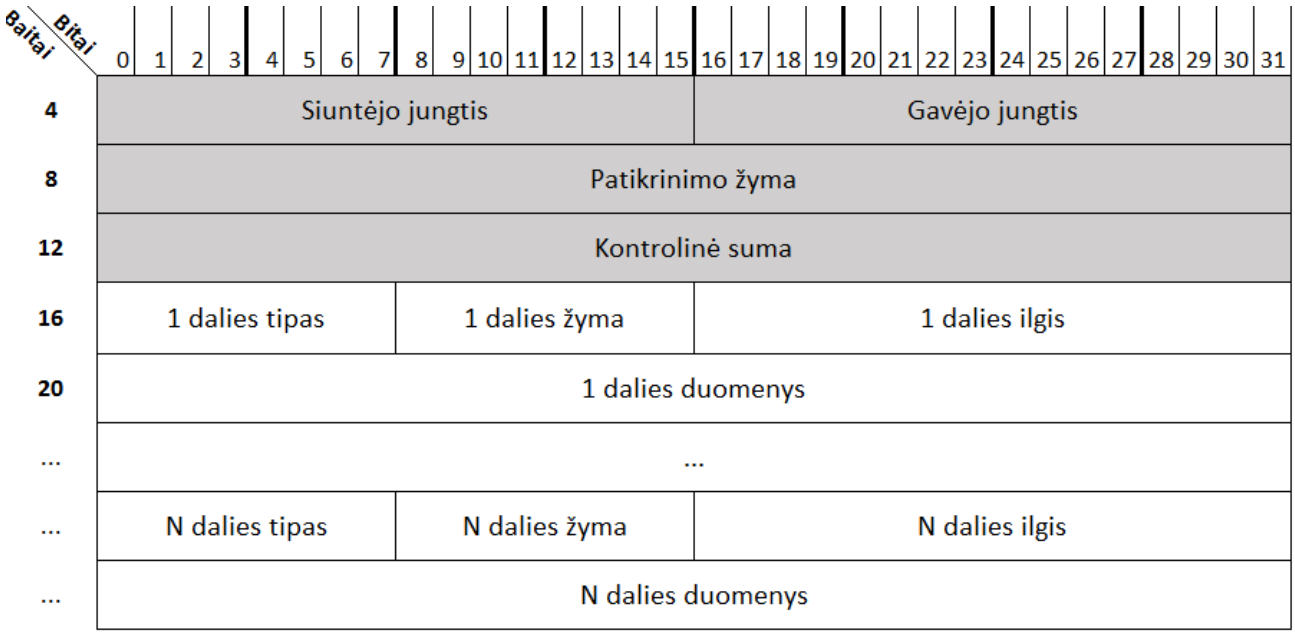
srautas yra ganėtinai svarbus tokiose žaidimuose. Taip pat FPS žaidimuose, žaidėjui išvengus kulkos dėl pradingusio duomenų paketo galima traktuoti kaip sėkmės faktorių.

MMORPG žaidimuose, TCP protokolas suteikia patikimumą, kad duomenų paketai nukeliaus žaidėjams ir taip visi veiksmai bus sukurti vienodai klientų pusėse. Taip pat šiuose žaidimuose kai kuriems veiksmams turi įtakos duomenų srauto sparta, pvz. žaidėjas laimės kovą, kuris pirmas sureaguos į veiksmus. Kaip rašo [SKR07] straipsnis, tokiems žaidimams yra reikalinga technologija, kuri sparčiai galėtų perdavinėti TCP paketus.

1.4. SCTP

SCTP, srauto kontrolės perdavimo protokolas (angl. *stream control transmission protocol*) [Hed11], panašus į TCP, orientuotas į patikimą duomenų perdavimą, buvo sukurtas stengiantis pašalinti TCP protokolo trūkumus, tokius kaip grūstis kanalo gale (angl. *head-of-line blocking*) ir pažeidžiamumas atsisakymo aptarnauti (angl. *denial of service*), dar kitaip žinomoms kaip „SYN-flooding“ atakomis. Taip pat protokolas turi panašių savybių kaip UDP - gali išskirti nepatikimą kanalą duomenų perdavimui, suteikia galimybę perduoti duomenis nerikiuojant juos eilės tvarka. SCTP yra daug jaunesnis protokolas nei TCP, pirmą kartą aprašytas 2000 metais, o standartizuotas IETF 2007 m. [Ste07].

SCTP duomenų paketas susideda iš antraštės ir duomenų blokų vadinamų dalimis (angl. *chunks*). SCTP duomenų paketo struktūra yra pavaizduota, 3 pav. Jame tamsesni laukai žymi bendrus antraštės laukus, kurie priklauso kiekvienam šio protokolo duomenų paketui. Antraštei priklauso tokie laukai kaip siuntėjo bei gavėjo jungtys, kontrolinė suma ir patikrinimo žyma (angl. *verification tag*). Pastaroji saugo atsitiktinį skaičių, kuris sugeneruojamas pirmą kartą užmezgus ryšį. Šis skaičius skirtas atpažinti duomenų paketus priklausančius vienai sesijai. Dalys gali saugoti apkrovos duomenis arba informaciją susijusią su protokolo valdymu. Jos susideda iš kelių laukų: dalies tipas, kuris nurodo kokia dalis yra siunčiama, dalies žyma, kuri saugo specifinę informaciją priklausomai nuo dalies tipo. Po to seka dalies ilgis išreikštas baitais ir dalies duomenų laukas, kuris taip pat saugo specifinę informaciją priklausomai nuo dalies tipo, į kurią galį įeitį ir pačios apkrovos turinys. Kelios dalys gali priklausyti vienam duomenų paketui, tačiau ne visos. Skirtingų, įvardintų dalies tipų yra 15, o nenaudojamų yra 240. Dalys yra skirstomos į tokius tipus kaip: apkrovos, ryšio užmezgimo, patvirtinimo, nutraukimo, klaidos ir kiti, kurie yra plačiai aprašyti straipsnyje [Ste07].



3 pav. SCTP duomenų paketo struktūra

Šis protokolas turi pora funkcijų, kurių neturi UDP ir TCP protokolai – kelių dažnių (angl. *multi-homing*) ir kelių kanalų (angl. *multi-streaming*) išskyrimas. Pastaroji funkcija suteikia keletą duomenų perdavimo kanalų vietoj vieno – priešingai nei TCP ar UDP. Pavyzdžiui TCP paketui nepasiekus gavėjo, kiti sekantys paketai lauks eilėje kol prarastasis paketas bus išsiųstas pakartotinai, taip susidaro kamštis gavėjo pusėje (grūstis kanalo gale). Per SCTP suteiktus kelis duomenų perdavimo kanalus, duomenys yra neužsilaikomi ir be trikdžių pasiekia gavėją.

Įvertinant SCTP protokolo privalumus iškylą klausimas, kodėl jis nėra plačiai naudojamas. Vienas paaiškinimas būtų toks, kad šis protokolas yra ganėtinai jaunas palyginus su TCP ir UDP protokolu. TCP yra naudojama kelis dešimtmečius, o pagrindiniai protokolo trūkumai pradėjo matytis po daugelį metų nuo išleidimo. Kai kuriuose straipsniuose [Jon06] svarstoma, kad SCTP trūkumai taip pradės matytis , kas užims taip pat daug laiko, kol trūkumai išlys į paviršių. Tačiau taip pat yra straipsnių [AIN+06], kuriuose rašoma, kad SCTP yra daug geresnis transporto protokolas nei TCP, nes pašalina to protokolo trūkumus, ypatingai suteikiant apsaugą nuo atsiskaitymo aptarnauti atakų.

Straipsnyje [Hed11] yra rašoma apie SCTP protokolo realizacijos pritaikymą internetiniuose žaidimuose. Jame svarstoma pagrindinė idėja, kaip optimizuoti internetinių žaidimų duomenų srautą ir sumažinti delsos laiką pasinaudojant specifinėmis protokolo savybėmis – dalinis patikimumas (angl. *partial reliability*), išrikiuotų ir neišrikiuotų duomenų paketų perdavimas, kelių kanalų išskyrimas.

Dalinis patikimumas tai tokia SCTP protokolo savybė, kuri suteikia taikomosioms programoms lankstumą leidžiant pasirinkti ar perduodant duomenys būtina užtikrinti jų pristatymą, ar

duomenų apkrova nėra svarbi ir paketų pristatymo užtikrinimas nėra būtinas. Taip pat duomenų paketams yra galimybė suteikti gyvenimo trukmę, kuriai pasibaigus paketas bus išimtas nepaisant ar būtų pristatytas gavėjui. Ši savybė yra svarbi taikomosioms programoms, kurios pastoviai generuoja duomenų paketus ir naujesnė informacija yra naudojama vietoje senos. Pritaikymas internetiniuose žaidimuose galėtų būtų toks, kai žaidėjo avataras pastoviai judėdamas generuoja naujas pozicijų koordinates ir jos yra siunčiamos žaidimo serveriui [AIN+06].

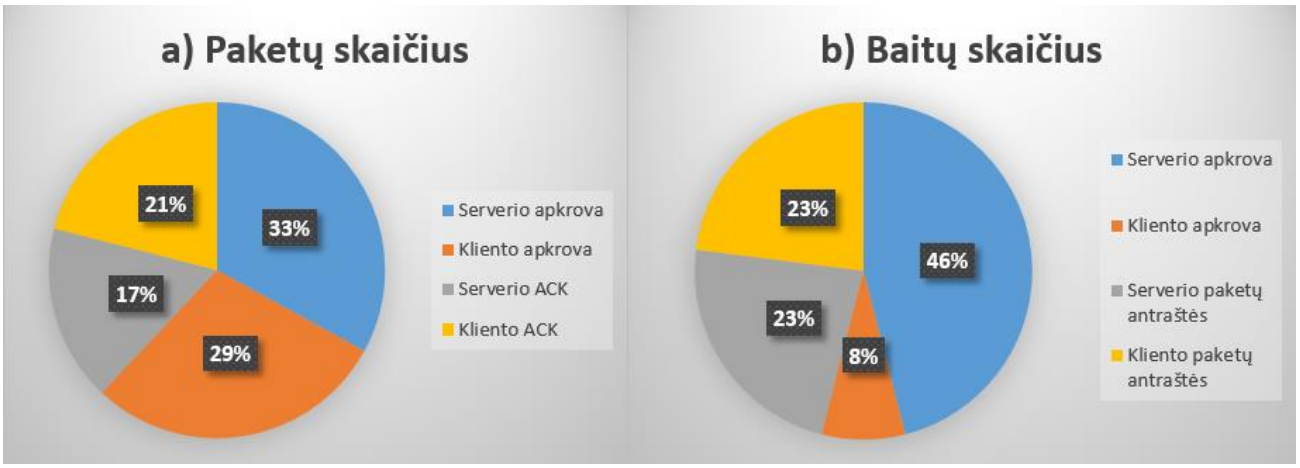
Išrikiuotų ir neišrikiuotų duomenų paketų perdavimas tai dar viena svarbi SCTP protokolo savybė, kuri suteikia galimybę pažymėti tam tikrus duomenų paketus, kurios norime siųsti be nustatytos eilės tvarkos. Taip paketai būna iš karto nuskaitomi, kai jie atvykstą į gavėjo kanalo galą. Šį savybė yra įgyvendinama išskiriant SCTP protokolui papildomą kanalą, skirtą perduoti duomenis nerikiuojant jų. Toks funkcionalumas yra naudingas internetiniams žaidimams, nes per kanalus galime išskaidyti nesusijusią informaciją turimus duomenų paketus, pavyzdžiui viename kanale siunčiame žaidėjų pozicijos atnaujinimus, o kitame kanale galime siųsti žaidėjų susirašinėjimo žinutes, ar perduoti žaidėjų pokalbius (angl. *voice-chat*) [AIN+06].

Tokios savybės prideda SCTP duomenų paketui papildomas dalis, kurios didina pačio paketo dydį. Taip pat protokolo standartiniai nustatymai sukelia didelę duomenų perdavimo delsą, todėl straipsnyje yra rekomenduojama šio protokolo nenaudoti internetiniuose žaidimuose [Hed11]. Tačiau yra taip pat rašoma, kad modifikuojant SCTP protokolo realizaciją būtų galima pasiekti geresnių duomenų perdavimo spartos rezultatus už TCP protokolo [PBP+09]. Modifikuota SCTP protokolo realizacija yra eksperimentinėje fazėje ir reiktų atlikti papildomus našumo testus norint įsivertinti šio protokolo pritaikymą internetiniuose žaidimuose.

SCTP yra palaikomas tokių operacinių sistemų kaip Linux, Solaris, FreeBSD, tačiau norint realizuoti šį protokolą Windows operacinėje sistemoje reikia naudotis trečiųjų šalių sprendimais, kaip SctpDrv kernel tvarkykle.

2. Duomenų paketų suspaudimo metodų analizė

Iš internetinių MMORPG žaidimų surinktų statistinių duomenų [CHH+15], buvo sudaryti grafikai, matomi 4 pav., juose matome, kad gan didelė dalis siunčiamų duomenų sudaro paketo TCP/IP antraštės (46%) ir TCP patvirtinimo paketai (38%). Taip pat MMORPG žaidimai dažniausiai išjungia Naglio algoritmą, kad duomenų paketai būtų išsiunčiami iš karto ir dėl to yra sukuriama daug nedideliu duomenų paketų, kurie yra siunčiami dažnai, ir jiems yra priskiriamos beveik vienodos antraštės. Skiriasi tam tikri laukai kaip kontrolinė suma, sekos numeris). Šiame scenarijuje galima būtų pritaikyti antraštės suspaudimo algoritmą.



4 pav. Internetinių žaidimų perduodamų duomenų TCP protokolo pridėtinė informacija

2.1. Duomenų paketų antraščių suspaudimo metodai

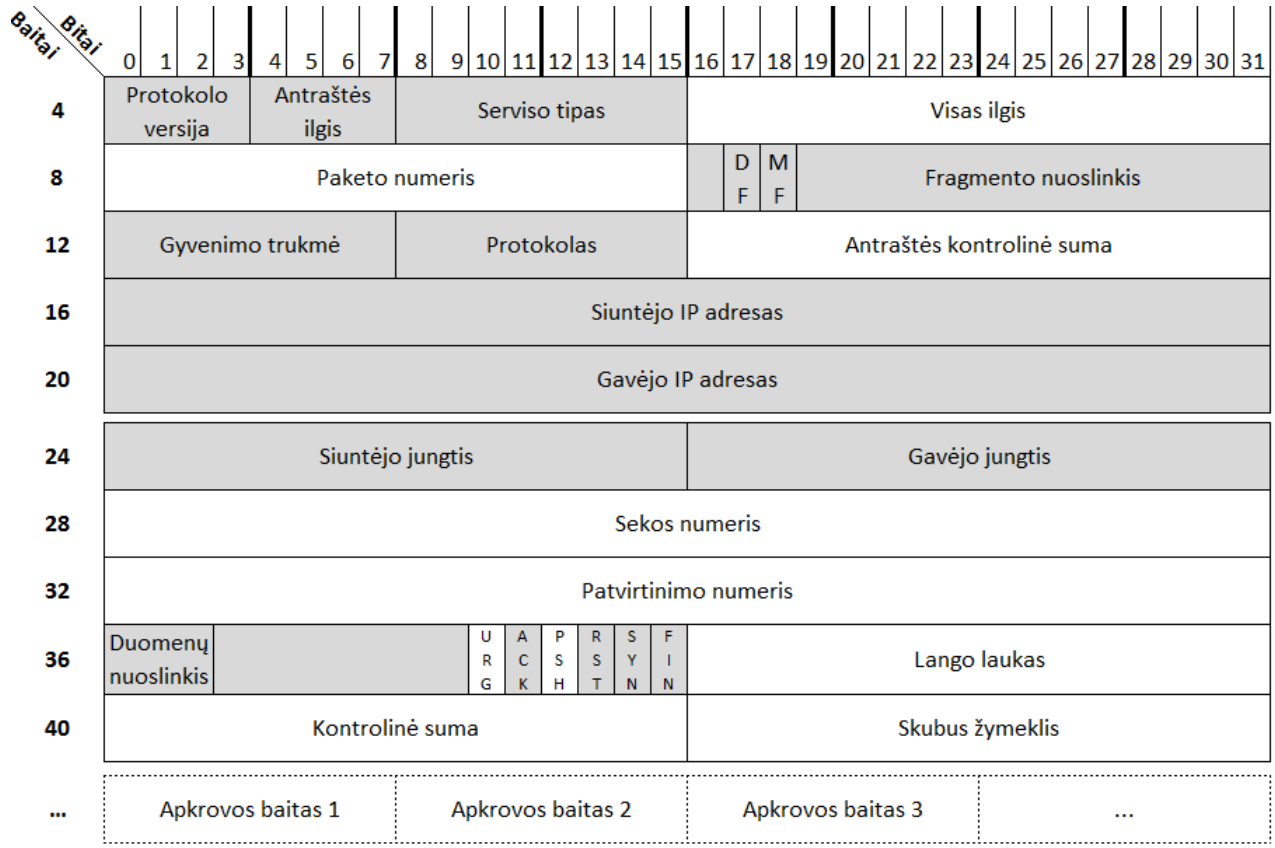
Egzistuoja keli transporto protokolų antraščių suspaudimo metodai. Pirmasis metodas skirtas suspausti TCP/IP antraštes buvo pristatytas Van Jacobson [Jac90]. Vėliau IPHC [DNP99] taip pat pridėjo galimybę suspausti IPv6 ir UDP antraštes. Tuo pačiu metu buvo pristatytas cRTP, kuris geba suspausti IP/UDP/RTP antraštes. Paskutinis suspaudimo algoritmas, ROHC [Bor01] buvo pristatytas 2001 metais, kuris suteikia apsauga nuo konteksto išsiskyrimo (angl. *context desynchcronization*), kuris dažniausiai nutinka bevieliuose tinkluose.

Dėl temos aktualumo buvo pasirinkta analizuoti Van Jacobson ir ROHC tinklo antraščių suspaudimo metodai. IPHC labiau orientuotas IPv6 antraščių suspaudimui, o cRTP - UDP duomenų paketo antraštėms.

2.1.1. Van Jacobson suspaudimas

Duomenų paketų antraščių suspaudimo sąvoką pirmą kartą pristatė Van Jacobson 1990 metais. Šia technologija buvo siekiama pagerinti TCP/IP duomenų paketo atsako laiką mažos spartos (300-19200 bps – bitų per sekundę) serijiniuose modemuose [Jac90].

Dažniausiai TCP/IP paketo antraštė sudaro 40 baitų – 20 baitų IP ir 20 baitų TCP interneto schemos dydžio. Van Jacobson pristatytas metodas siūlo išmesti pasikartojančius antraštės laukus jei jie nesikeičia. Paprastai išmesti tam tikrus laukus iš antraščių negalime nes kiekvienas jų saugo tam tikrą svarbią informaciją. Tačiau, kai TCP protokolu yra sukuriamas kanalas, tipiniais atvejais per jį yra perduodami šimtai ar tūkstančiai paketų. Iškylą klausimas – kiek tą pati informacija, kuri nesikeičia antraštėje, yra perduodama su kiekvienu paketu. Kaip matome 5 pav., beveik puse antraštės laukų saugo nekintančią informaciją – pilkesni laukai. Jeigu siuntėjas ir gavėjas saugotų prieš tai gauto duomenų paketo antraštę, tuomet siuntėjas galėtu išsiųsti tik antraštės pasikeitimus, o gavėjas galėtų atkurti šią antraštę pasinaudojant išsaugota, prieš tai gautą, pilną antraštę. Taip būtų galima sutaupyti maždaug 20 baitų tipinės TCP/IP paketo antraštės, jei puse laukų nesikeistų.



5 pav. Perduodami TCP/IP antraštės laukai, kurie nekinta

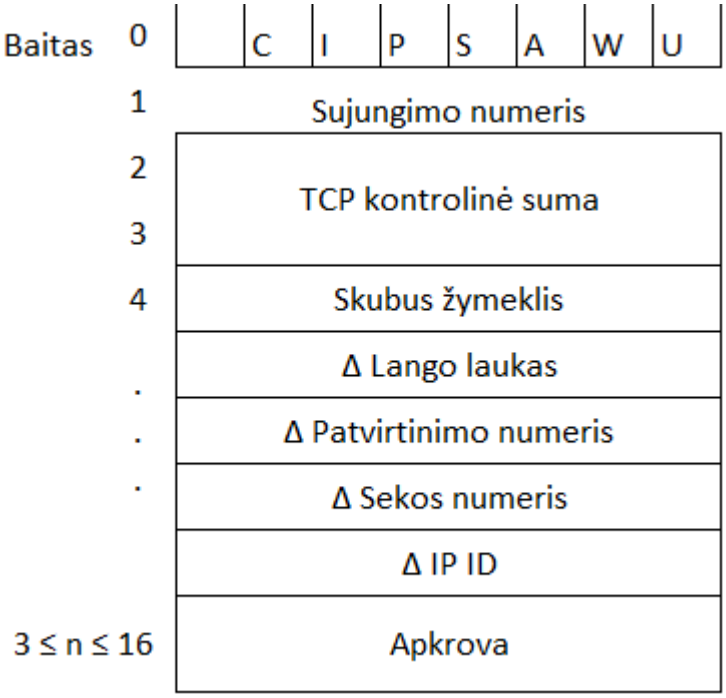
Dar galima sutaupyti pora baitų nesiunčiant viso duomenų paketo dydžio, jei ši informacija yra žinoma kliento pusėje. Po to sekančius du baitus galima sutaupyti nuo IP antraštės kontrolinės sumos, nes ji skirta patikrinti ar visi IP antraštės laukai pasiekė gavėją, o šiuo atveju būtų siunčiamas tik duomenų paketo numerio laukas.

Iš viso lieka 16 baitų nuo tipinio 40 baitų TCP/IP paketo. Visi likę antraštės laukai anksčiau ar vėliau keisis, tačiau jie keičiasi ne visi iš karto. Pavyzdžiui per FTP duomenų perdavimą keičiasi tik paketo, sekos numerių ir kontrolinės sumos laukai siuntėjo į gavėjo pusę. Siunčiant paketus atgal, iš gavėjo siuntėjui, keičiasi tik paketo numeris, patvirtinimo žyma, kontrolinė suma ir lango laukas. Turint kopija paskutinio gauto paketo, siuntėjas gali išsiaiškinti kurie laukai keičiasi ir naudojant bitų maskavimo technologiją (angl. *bitmask*) pridėti antraštėje informaciją saugančia kokie laukai keičiasi.

Jei siuntėjas siųs tik tuos laukus kurie keičiasi, antraštė susispaus iki vidutiniško 10 baitų dydžio. Norint dar labiau sutaupyti baitus reikia atkreipti dėmesį į kaip tam tikri laukai keičiasi. Paketo numeris dažniausiai keičiasi didėjimo seka, per vienetą kiekvienam paketui (1, 2, 3…). Skirtumas tarp paketų numerių yra mažas – mažiau negu 256 bitai (vienas baitas), o dažniausiai skiriasi tik vienetu. Tas pats galioja ir sekos numeriui, iš siuntėjo pusės jis padidės vienetu ir dar prisidės skaičius gautų

paketų iš gavėjo nuo praeito išsiųsto paketo. Šiuose dviejose laukuose perduodant pasikeitimų informaciją, ant kiek paketo ir sekos numeris padidėjo, vietoj pačių numerių, galima būtų sutaupyti dar 3 - 4 baitus.

Taip yra suspaudžiama tipinė 40 baitų TCP/IP antraštė net iki 5-6 baitų. Specifinėse situacijose baitų skaičius gali siekti minimaliai 3 baitus. 6 pav. matome kokie duomenys būtų perduodami suspaustoje antraštėje: kontroliniai bitai, sujungimo numeris (angl. *connection number*) – saugantis informacija, kuris duomenų paketas buvo išsiųstas prieš tai, tam kad gavėjas žinotų kurią saugoma kopiją praeitos antraštės naudoti atspaudžiant antraštę, TCP kontrolinė suma – skirta nustatyti paketo apkrovos integralumą ir taip pat jau prieš tai minėti TCP protokolo duomenų paketo antraštės laukai.



6 pav. Van Jacobson suspaustos TCP/IP antraštės turinys

Taikant šį suspaudimo algoritmą ant senų serijinių ir telefoninio ryšio (angl. *dialup*) modemų buvo pastebėtą, kad iškyla nemažai klaidų, nes paketo apkrovos integralumo tikrinimo mechanizmas remiasi kontrolinės sumos laukais, o šios technologijos atspaudimo mechanizmas galėdavo atkurti ir pažeistus paketus, kai dalis duomenų paketų apkrovos dėl blogo ryšio dingdavo, o atspaudimo mechanizmas atkurdavo tariamai korektišką kontrolinę sumą.

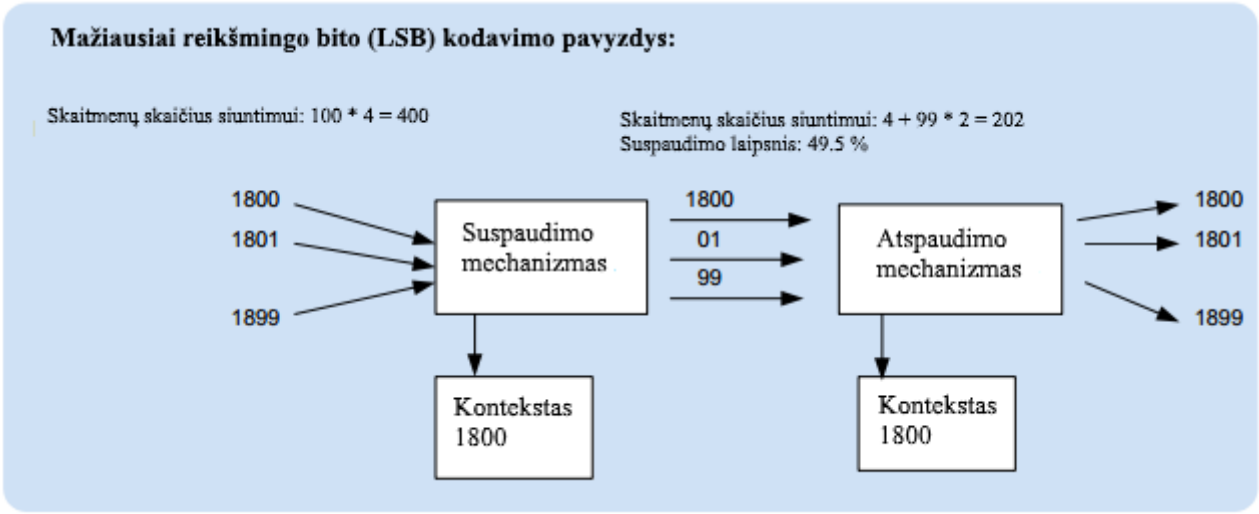
Šio suspaudimo algoritmo galimoms realizacijoms yra parašytas C kalbos programinis kodas (nuoroda: < ftp://ftp.rfc-editor.org/in-notes/rfc1144.pdf>).

2.1.2. ROHC

ROHC – tai tvirtas antraščių suspaudimo karkasas (angl. *robust header compression*), skirtas suspausti IP, UDP, RTP ir TCP internetinių paketų antraštes. Šis karkasas buvo pristatytas kaip naujas antraščių suspaudimo standartas [Bor01].

ROHC suglaudina tipinę 40 baitų antraštę iki 1 - 3 baitų, priskiriant suspaudimo ir išskleidimo mechanizmą, prieš kanalą ir kanalo gale. Suspaudimo mechanizmas suspaudžia antraštę iki kelių baitų, o atspaudimo mechanizmas iš tų kelių baitų atkuria pilna antraštę.

Pagrindinis skirtumas ROHC suspaudimo metodo nuo Van Jacobson yra tas, kad pastarajame yra išimami pasikartojantys laukai ir yra ieškoma kaip tie laukai keičiasi, jiems priskiria deltą laipsnį ant kiek laukai keitėsi, vietoj pasikeitusios reikšmės, o ROHC naudoja mažiausiai reikšmingo bito algoritmą - LSB (angl. *least significant bit*), kuris pasižymi didesniu suspaudimo laipsniu. 7 pav. matome LSB kodavimo pavyzdį, kuriame yra užkoduojami skaitmeniniai duomenys. Jei atspaudimo mechanizmui yra žinomas duomenų kontekstas, šiuo atveju 19 amžiaus metų masyvas, tai perduodant tik dalį duomenų, atspaudimo algoritmas sugeba atkurti pilnus duomenis, tokia pat forma, kurią jie turėjo prieš suspaudimą.



7 pav. LSB kodavimo pavyzdys

Naudojant LSB algoritmą, reiktų perduoti ir duomenų kontekstą, ten kur jis nebūtų žinomas, ko nereikalauja Van Jacobson suspaudimo algoritmas, tačiau šiuo karkasu galima būtų suspausti ne tik TCP antraštes, tačiau ir UDP.

Didesniam suspaudimo laipsniui gauti, duomenų paketai yra suklasifikuojami į srautus dar prieš suspaudimą. Taip klasifikuojant yra išvengiama pasikartojantys duomenų laukai esantys tarp siunčiamų duomenų paketų. Klasifikacijos algoritmas nėra pristatytas ROHC karkase, jį turi pridėti programos kūrėjas, pasirinkus, kurį transporto protokolą jis naudos. TCP/IP protokolo antraštės laukų klasifikacija yra plačiai aprašyta straipsnyje [Wes13].

Yra platinama atviro kodo ROHC karkaso biblioteka (nuoroda: <https://rohc-lib.org/>), skirta programinės įrangos kūrėjams, norintiems sumažinti duomenų srautą tarp kuriamų taikomųjų programų. Ši biblioteka palaiko tokias platformas kaip Linux, BSD, Windows ir Android.

2.2. Duomenų paketų apkrovos suspaudimo algoritmai

Dėl temos aktualumo buvo pasirinkta nagrinėti LZW ir Huffman duomenų suspaudimo algoritmus. Abu algoritmai buvo naudojami suspaudžiant duomenų paketus transporto protokoluose [Kle91][Jac90].

2.2.1. LZW

Lempel-Ziv-Welch yra universalus nenuostolingas glaudinimo algoritmas. Pristatytas Terry Welch, kaip patobulinimas jau egzistuojančio LZ78 algoritmo [Wel84]. Algoritmą yra paprasta realizuoti ir jis yra populiarus tarp įvairių technologinių sprendimų: paveikslų suspaudimas – GIF, teksto taikomojoje programoje – PDF ir kiti.

Duomenys yra suslegiami ieškant vienodų sekų, vadinamų frazėmis. Rastos sekos yra išsaugomos lentelėje ir joms yra skiriami trumpesni 8 bitų raktai.

Abiejuose straipsniuose [Kle91][Jac90] rašoma apie algoritmo suspaudimo laipsnio naudingumą, tačiau minima, kad suspaudimo skaičiavimo laikas yra ganėtinai didelis.

Šio suspaudimo algoritmo yra galimos realizacijos parašytos įvairių programavimo kalbų (nuoroda: <http://rosettacode.org/wiki/LZW_compression >).

2.2.2. Huffman

Huffman suspaudimo algoritmas [Huf07] yra paremtas simbolių dažnumu duomenyse, kuriuos norime suspausti. Pritaikant Huffman algoritmą, dažnesniausiems simboliams yra suteikiama mažiau bitų, o retesniems daugiau. Taip yra sukuriamas Huffman medis, kuris yra naudojamas duomenų suspaudimui. Jei simbolių dažnumas nėra žinomas atspaudžiant duomenis, tai Huffman medis turi būti perduodamas atkodavimo mechanizmui kartu su suspaustais duomenimis. Huffman algoritmas buvo vienas pirmesnių geriausių suspaudimo algoritmų, sukurtas 1952 metais.

Iš išanalizuotų duomenų srauto modelių internetiniuose žaidimuose [CHH+15] yra nustatyta, kad daugiausia duomenų paketų yra siunčiama dėl žaidėjų pozicijų atnaujinimo. Huffmano suspaudimo algoritmas tokiems duomenimis nebūtų idealus, nes simbolių dažnumas žaidėju koordinatėse nėra žinomas iš karto dar neturint tų koordinačių, todėl Huffman medis turėtų būti siunčiamas kartu su koordinatėmis. Medžio dydis gali atsverti suspaustų duomenų dydį. Tačiau šį suspaudimą galima būtų taikyti paketuose su didesniu duomenų skaičiumi.

Šio suspaudimo algoritmo realizacijos yra sukurtos ant įvairių programavimo kalbų (šaltinis: <http://rosettacode.org/wiki/Huffman_coding>).

2.2.3. Huffman ir LZW suspaudimo algoritmų palyginimai

Straipsnyje [JK14] rašoma apie Huffman ir LZW suspaudimo algoritmų našumo tyrimus, pritaikant algoritmų realizacijas belaidžių sensorių tinkluose (angl. *wireless sensor networks*). Našumas yra vertinamas pagal suspaudimo laipsnio ir skaičiavimo laiko, į kurį įeina duomenų suspaudimas ir atspaudimas, santykiu. Tyrime naudojami realūs duomenys surinkti iš belaidžių sensorių, tokie kaip: temperatūros, drėgmės rodmenys ir tam tikri tekstiniai duomenys. Šių duomenų tipai yra tam tikra prasme panašūs į internetinių žaidimų siunčiamų duomenų tipus, pavyzdžiui: žaidėjų pozicijų atnaujinimų masyvas, kurį sudaro sveikieji skaičiai arba slankiojo kablelio skaičiai, kas atitinka temperatūros ir drėgmės duomenų masyvą. Taip pat tyrime buvo vertinama dvigubo suspaudimo technologija, kurioje seka vienas po kito Huffman ir LZW suspaudimo algoritmai – HLZ, ir atvirkščia tvarka – LZH. Tyrime naudojami duomenys buvo sugrupuoti į 200, 400, 600 ir 800 bitų grupes, tam, kad būtų įvertinta suspaudimo našumo priklausomybė nuo duomenų kiekio.

Suspaudimo laipsnio tyrimo rezultatai parodė, kad stipriausiai buvo suspausti 200 bitų grupės temperatūros duomenys, naudojant Huffman algoritmą, tačiau suspaudimo laipsnis krinta kai

duomenų daugėja. Taip atsitinka todėl, kad Huffman medis didėja kartu su duomenų skaičiumi. Kuo ilgesnės Huffman medžio šakos tuo ilgesnis tampa Huffman kodas. LZW suspaudimo algoritmas parodė menką suspaudimo laipsnį, todėl, kad šis algoritmas suspaudžia duomenis bitą po bito, kas yra neefektyvu tokio tipo duomenyse, nes jie yra jau ir taip sudėlioti į bitų grupes. LZW suspaudimo laipsnis pradeda didėti nuo 800 ir daugiau bitų tekstinių duomenų, taip nutinka todėl, kad pradeda kartotis žodžiai tekste, kurie atitinka sekas šio algoritmo lentelėje. Dvigubame suspaudime LZH algoritmų seka parodo didesnį suspaudimo laipsni už HLZ seką. Taip yra todėl, kad Huffman algoritmas išdėlioja suspaustus duomenis specifine tvarka, kuri nėra optimali LZW algoritmui. LZW suspaustus duomenis pateikia išeiga su dideliu pasikartojančių reikšmių skaičiumi.

Skaičiavimo laiko tyrimo rezultatai parodė, kad visų duomenų suspaudimų skaičiavimas buvo atliktas greičiausiai Huffman algoritmo. Taip yra todėl, kad Huffmano suspaudimo algoritmas yra paprastesnis už LZW algoritmą. Nors LZW duomenų atspaudimo skaičiavimas yra atliekamas greičiau už Huffman, dėl to, kad suspaustuose duomenyse yra ieškoma sekų atitikimų pagal lentelę, o kur Huffman algoritmas atspaudimui tikrina bitą po bito. Tačiau LZW suspaudimo algoritmo skaičiavimo laikas yra ženkliai didesnis lyginant su Huffman algoritmu.

Straipsnio autorius pateikia išvadas, kad Huffman algoritmas pasižymi didesniu našumu už LZW algoritmą. Huffman suspaudimo laipsnis pasiekia vidutiniškai 43 procentus, šį skaičių galima dar padidinti 9 procentais pasinaudojus dvigubą suspaudimo seka pradedant nuo LZW ir po to Huffman, tačiau tuomet ženkliai didėja skaičiavimo laikas. Žinoma LZW suspaudimo algoritmo nereikia nuvertinti, nes juo galima būtų pasinaudoti perduodant tekstinius duomenis internetiniuose žaidimuose, tokius kaip susirašinėjimai tarp žaidėjų ir perspėjimo žinutės žaidime.

2.2.4. SIMD komandos

SIMD – viena komanda, keletas duomenų elementų (angl. *single instruction, multiple data*), tai architektūra, kuri leidžia viena procesoriaus komanda lygiagrečiai apdoroti keletą vienodo tipo supakuotų duomenų elementų. SIMD komandos [DLN15] buvo pristatytos 1999 m. Pentium III serijos procesoriuose. Šios komandos skirstomos į 4 kategorijas:

1. SIMD – komandos vienetinio tikslumo duomenims su slankiuoju kableliu (SPFP komandos);
2. Papildomos SIMD komandos pilniems duomenims;
3. Podėlio (angl. *cache*) valdymo programos;

4. Procesoriaus būsenos išlaikymo ir atstatymo komandos.

SIMD komandų naudojimas leidžia žymiai pagreitinti slankųjį kablelį naudojančių programų greitį, nes viena procesoriaus komanda lygiagrečiai yra apdorojami keli duomenų elementai. Atlikus kiekvieną tokią komandą yra gaunamas keletas skaičiavimo rezultatų. Viena SIMD komanda su slankiu kableliu tuo pat metu gali apdoroti keturis vienetinio tikslumo 32 bitų skaičius su slankiu kableliu, kurie yra vadinami SPFP (angl. *single precision floating point*) duomenų elementais. Komandos su SPFP duomenims naudoja 8 naujus 128 bitų registrus (XMM0,...,XMM7) [JND15], kuriuose laikomas naujas informacijos tipas – 4 paeiliui išdėlioti 32-jų bitų skaičiai su slankiu kableliu. Šie registrai išskirtinai naudojami darbui su duomenimis. SIMD leidžia 2 duomenų apdorojimo būdus – lygiagretų ir skaliarinį.

SIMD komandos su slankiu kableliu taikomos: aritmetiniams veiksmams, kvadratinių šaknų radimui, apytikrio sprendimo radimui (priartinimui), minimumo ir maksimumo radimui, duomenų perkėlimui, ženklų kaukės sukūrimui, palyginimui ir ženklų kaukės sukūrimui, loginėms operacijoms, palyginimui ir EFLAGS nustatymui, duomenų keitimui.

Straipsnyje [DLN15] yra rašoma apie šios technologijos panaudojimą išrūšiuotų skaičių masyvų suspaudimuose. Išrūšiuoti 32 bitų skaičiai yra suspaudžiami pasinaudojus SIMD komandomis ir FastPFOR algoritmu [VS14]. Šis algoritmas siekia kaip įmanoma dažniau išnaudoti SIMD komandas skaičių masyvo suspaudimui pasitelkus bitų maskavimo technologiją.

SIMD komandų panaudojimas yra detaliai aprašytas internetiniame puslapyje (šaltinis: <https://www.kernel.org/pub/linux/kernel/people/geoff/cell/ps3-linux-docs/CellProgrammingTutorial/BasicsOfSIMDProgramming.html>).

3. Metodų, kurie mažina duomenų paketų perdavimo dažnį, analizė

Buvo išanalizuoti populiarūs internetiniai žaidimai, kurie pasižymi įvairiais virtualaus pasaulio, žaidėjų, išskaidymo metodais matomais 2 lentelėje. Pasirinkti įvairių metų internetiniai žaidimai tam, kad matytųsi, jog kai kurie metodai yra išlikę iš seniau ir yra naudojami iki šiol internetiniuose žaidimuose. Taip pat pastebėta, kad yra bandoma realizuoti naujus metodus („Tree of Savior“ žaidimas realizavo žaidėjų užslėpimo technologiją).

2 lentelė. Internetinių žaidimų metodai, kurie mažina duomenų paketų perdavimo dažnį

Žaidimo pavadinimas	**Žanras**	**Išleidimo metai**	**Naudojami metodai**
Quake 3	FPS	1999 m.	Apribotos virtualios erdvės
Ragnarok online	MMORPG	2002 m.	Apribotos virtualios erdvės, platūs virtualieji pasauliai
Lineage 2	MMORPG	2003 m.	Platūs virtualieji pasauliai
World of Warcraft	MMORPG	2004 m.	Apribotos virtualios erdvės, platūs virtualieji pasauliai
Tera online	MMORPG	2011 m.	Apribotos virtualios erdvės, platūs virtualieji pasauliai, kanalų išskyrimas
Blade and Soul	MMORPG	2012 m.	Apribotos virtualios erdvės, platūs virtualieji pasauliai, kanalų išskyrimas
Maple Story 2	MMORPG	2015 m.	Platūs virtualieji pasauliai, kanalų išskyrimas
Tree of Savior	MMORPG	2016 m.	Apribotos virtualios erdvės, platūs virtualieji pasauliai, kanalų išskyrimas, žaidėjų užslėpimas

3.1. Apribotos virtualios erdvės

Yra manoma, kad pirmas MMORPG žaidimas pradėjęs naudoti šį metodą buvo „The Realm Online“ išleistas 1996 metais ir nuo to laiko šio metodo realizacija internetiniuose žaidimuose tapo dažnas atvejis. Žaidėjų išskaidymas į apribotas virtualias erdves, dar žinomoms kaip „Instance dungeon“ [SDM09], sumažina duomenų perdavimo skaičių tarp greta esančių žaidėjų. Priklausomai nuo žaidimo, žaidėjai norintys gauti specifinių artifaktų ar kitų privalumų, kuriu paprastai nėra galimybės gauti, privalo išsiskirstyti į mažas grupes, dažniausiai 5-10 žaidėjų, taip gaunant galimybę patekti į tam tikras virtualaus pasaulio vietas. Kelios grupės žaidėjų gali patekti į tas pačias vietas, tačiau tos grupės viena kitos nematys, visi veiksmai apribotuose vietose yra atliekami paraleliai.

Šis metodas dažnai sukelia žaidėju nepasitenkinimą žaidimu, nes žaidėjai būna priversti ieškoti komandos narių, kurie norėtų visi eiti į vieną iš apribotų virtualių erdvių. Taip pat tokiose erdvėse yra sugaištama nemažai laiko, todėl didelę laiko dalį, žaidėjų grupė atsiriboja nuo visų kitų žaidėjų esančių žaidime. Atsisakyti eiti į tokias erdves dažniausiai būna nenaudinga, nes juose gauti artifaktai, priklausomai nuo žaidimo, smarkiai įtakoja tolimesnius žaidėjų veiksmus. Tačiau yra potenciali galimybė taikyti šį metodą be didelių apribojimų, kad žaidėjai galėtų mėgautis žaidimu nepraleidžiant daug laiko tokiuose erdvėse.

3.2. Platūs virtualieji pasauliai

Jau senai šis metodas buvo taikomas internetiniams žaidimams – kuo didesnis virtualus pasaulis tuo didesnė tikimybė, kad viena didelė grupė žaidėjų išsiskaidys į mažesnes grupes po visą virtualų pasaulį, to pasėkoje siunčiamų duomenų paketų skaičius mažėja, nes žaidėjai atitolsta vienas nuo kito, o serveris neprivalo siųsti žaidėjo veiksmų tiems žaidėjams, kurie jo nemato. Tačiau virtualiuose pasauliuose dažniausiai būna pagrindinės susitikimų vietos, tokios kaip pagrindiniai žaidimo miestai, kuriuose greta esančių žaidėjų skaičius išlieka ganėtinai didelis.

Šiuo metodu pasižymi daug MMORPG žanro žaidimų įskaitant: „World of Warcraft“, „Lineage 2“, “Tree of Savior“, „Ragnarok online“, „Blade and Soul“, „Tera online“. Pastarajame šis metodas yra ypatingai suderintas su tam tikromis žaidimo funkcijomis, kaip artifaktų pardavimas, avatarų kovos ir virtualaus pasaulio teritorijų užėmimas. Žaidėjui yra suteikiama galimybė pasinaudoti šiomis operacijomis iš bet kurios virtualaus pasaulio vietos, pavyzdžiui, žaidėjo avataras gali parduoti savo artifaktus kitiems žaidėjams per aukcioną, kuris yra iškviečiamas per meniu funkciją – taip

žaidėjas neprivalo keliauti į virtualaus pasaulio miestus ir ieškoti didesnės grupės žmonių, kuriems reiktų reklamuoti savo avataro parduotuvę. Suteikiant žaidėjui tokias žaidimo funkcijas atlikti iš bet kurios virtualaus pasaulio vietos pašalina būtinybę žaidėjo avatarui ieškoti didesnės grupės žaidėjų. Toks šio metodo suderinamumas su pagrindinėmis žaidimo funkcijomis, padidina žaidėjų išskaidymo tikimybę virtualiame pasaulyje.

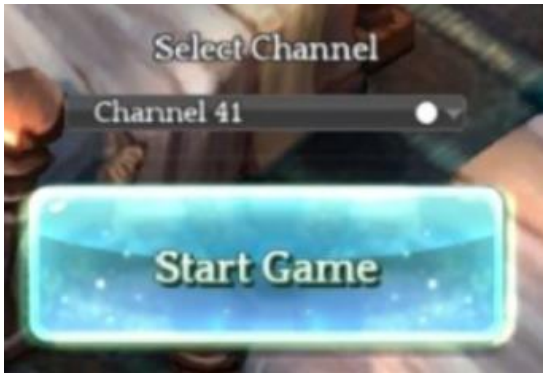
3.3. Žaidimo kanalų išskyrimas

Internetiniuose žaidimuose duomenų srauto skaidymas prasidėjo dėl žaidėjų pasiskirstymo per regionus. Žaidėjams esantiems tam pačiam regione kaip ir žaidimo serveris delsos laikas yra mažesnis nei žaidėjams kitose regionuose, dėl to, kad duomenų paketai turi nukeliauti trumpesnį kelią iki žaidimo serverio ir atgal. Taip atsirado nelygiavertiškumas tarp žaidėjų - žaidėjai su mažesne delsa buvo pranašesni už kitus. Todėl buvo išskiriami papildomi žaidimo serveriai, kaip matome 8 pav., dažniausiai vienas serveris per regioną. Tipiniame atvejui žaidėjui startavus žaidimo klientą jo yra prašoma pasirinkti, kuriame regione norėtu žaisti, taip yra išskaidomas visas žaidėjų srautas į skirtingus regioninius serverius. Žaidėjai skirtinguose serveriuose visados žaidžia atskirai, tarsi žaistų skirtingą žaidimą.

8 pav. Žaidimo serverių pasiskirstymas per regionus
(paveikslas paimtas iš „Tree of Savior“ internetinio žaidimo)

Tačiau kai žaidėjų skaičius išauga iki tam tikro skaičiaus, dėl kurio žaidimo serveris nepajėgia aptarnauti visų žaidimų klientų ir taip atsiranda didelė žaidimo delsa, mažėja žaidėjų pasitenkinimas

žaidimu. Dabar yra dažnai taikoma žaidimo kanalų išskyrimo technologija. Kiekvienas kanalas dažniausiai turi savo atskirus virtualius serverius ir kiekvienas iš jų turi atitinkamą žaidėjų skaičių, kurį gali aptarnauti be didelės žaidimo delsos. Žaidėjų skaičiui augant atsiranda daugiau žaidimo kanalų. Buvo pastebėta, kad MMORPG žaidime „Tree of Savior“ išleistam 2016 metais, kanalų skaičius siekė net iki 41, kaip matome 9 pav. Žaidimo kanalai prideda žaidėjams papildoma veiksmą, kurį jie turi atlikti – startavus žaidimą ir pasirenkant regioninį serverį, žaidėjas papildomai turi pasirinkti ir žaidimo kanalą, kuriame žais. Priešingai nei regioniniame serveryje, žaidėjo progresas žaidime išlieka vienodas per visus žaidimo kanalus. Tai reiškia, kad žaidėjui perėjus į kitą žaidimo kanalą jis nieko nepraranda (žaidėjo avataras su visais artifaktais išlieka žaidime).



9 pav. Ypatingai didelis žaidimų kanalų skaičius internetiniame žaidime „Tree of Savior“ (paveikslas paimtas iš „Tree of Savior“ internetinio žaidimo)

Šios technologijos trūkumas yra toks, kad priverčia žaidėjus patiems išsiskirstyti per visus žaidimo kanalus. Jei žaidėjas norės žaisti kartu su draugais, visi jie turės iš karto susitarti kuriame kanale žais. Tačiau ši technologija praverstu nenumatytuose scenarijuose, kai žaidėjų skaičius neplanuotai padidėtų ir reiktų skubios išeities gebėti aptarnauti visus žaidėjus.

3.4. Žaidėjų užslėpimas

Internetiniame MMORPG žaidime „Tree of Savior“, išleistam 2016 metais, buvo pastebėta nauja technologija, kuri mažina duomenų perdavimą tarp žaidėju juos užšelpiant vienus nuo kitų. Užslėpimo metodas yra nepublikuojamas ir neaiškus, pagal ką yra klasifikuojamos žaidėjų grupės kurios yra užslepiamos viena nuo kitos. Šis metodas buvo pastebėtas dėl netinkamo jo realizavimo –

virtualioje erdvėje pelyte paspaudus ant tariamai tuščios vietos, buvo išmetamas sąrašas su žaidėjų vardais, kaip matome 10 pav., kurie tariamai buvo užslėpti pačio žaidimo.

10 pav. Žaidime aptiktas užslėptų žaidėjų sąrašas
(paveikslas paimtas iš „Tree of Savior“ internetinio žaidimo)

Šis metodas kėlia nepasitenkinimą tarp žaidėjų, kildavo klausimų, kodėl būtent jie buvo užslepiami ir kaip juos atslėpti jei yra norima žaisti kartu su jais. Tačiau neverta nuvertinti šio metodo, nes jis pateikia įdomią idėją - užslėpti žaidėjus. Tinkamai realizavus ją, pavyzdžiui, aptinkant neveiksnius žaidėjus, kurių avatarai ilgą laiką stovi vienoje vietoje ir neatlieka jokių veiksmų, jiems būtų galima siųsti mažiau duomenų paketų, užslepiant aktyvius žaidėjus, arba tam tikrus jų veiksmus, iki tol kol neveiksnus žaidėjas sugrįš atgal į žaidimą ir taps veiksniu.

Iš [CHL05] straipsnyje pateiktų statistinių duomenų yra žinoma, kad „ShenZhou Online“ internetiniame žaidime daugiausia žaidėjų yra neveiksnūs, o straipsnyje [VM09] yra teigiama, kad MMORPG žaidimuose beveik puse žaidėjų būna neveiksnūs viso savo sugaišto žaidime laiko. Taip pat teigiama, kad žaidėjai būna dažniausiai neveiksnūs didžiausiuose virtualaus pasaulio miestuose, kur žaidėjų skaičius yra didžiausias. Tokiu atveju, žaidėjų užslėpimas neveiksniems žaidėjams atneštų ganėtinai reikšmingą naudą optimizuojant duomenų srautą.

4. Kiti duomenų srauto optimizavimo būdai

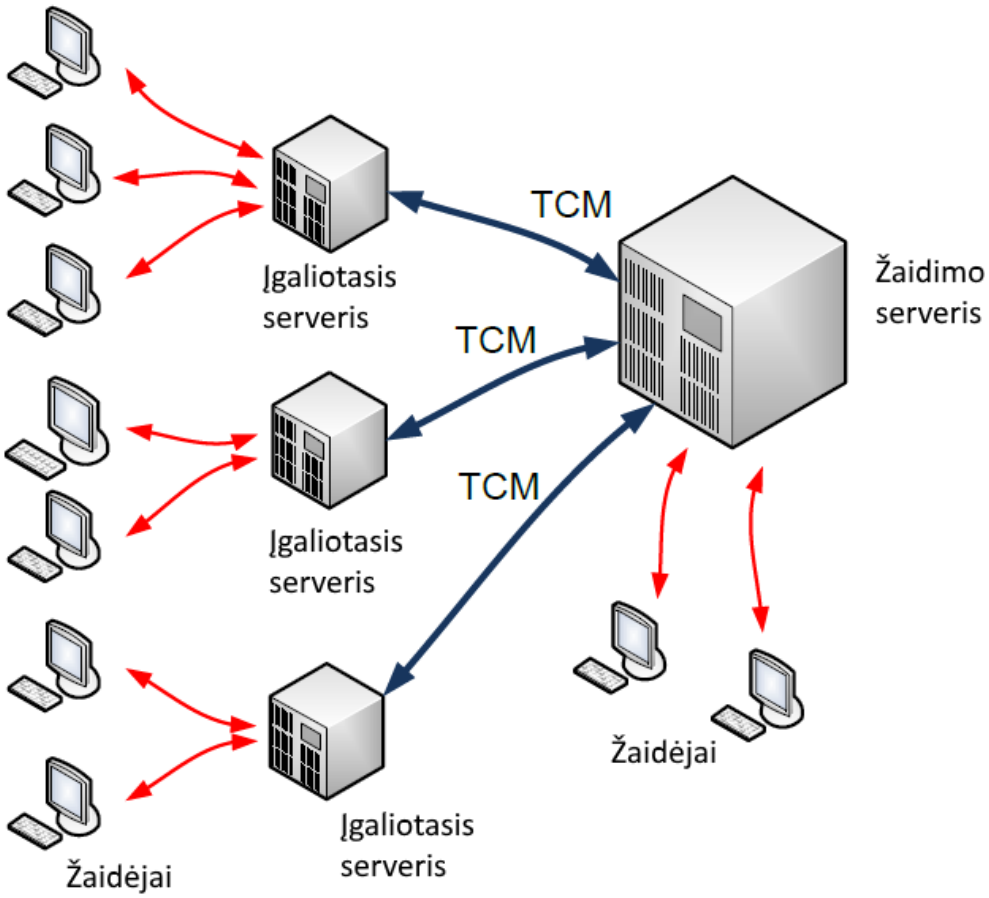
Telekomunikacijose į duomenų srauto optimizavimą taip pat įeina ir tuneliavimo bei tankavimo (angl. *multiplexing*) technologijos.

Tuneliavimas – tai procesas, kuomet vieno kompiuterinio tinklo protokolo paketo informacija yra „pernešama“ kitame pakete. Šis procesas dažnai yra vadinamas enkapsuliacija. Paketų seka tarp dviejų galinių tinklo taškų vadinama tuneliu todėl, kad po to, kai enkapsuliuota informacija yra pašalinama, nutolę taškai tarsi tiesiogiai, be „tarpininkų“ keičiasi informacija vienas su kitu. Tuneliavimo pagalba vienu protokolu galima „pernešti“ kito protokolo paketus, tačiau enkapsuliuoti paketai nebūtinai yra užkoduoti. Enkapsuliacija taip pat prideda maršrutizavimo protokolo informaciją šalia kito tipo informacijos. Tai yra svarbu todėl, kad IP paketas, keliaudamas per eilę Interneto maršutizatorių, visada turi turėti gavėjo adresą, pagal kuri maršrutizatorius nusprendžia, kaip paketas turi keliauti toliau.

Tankinimas - kelių duomenų perdavimo kanalų sujungimas į vieną sutankintą kanalą. Kanalams sujungti ir išskaidyti naudojamas tankintuvas.

4.1. TCM

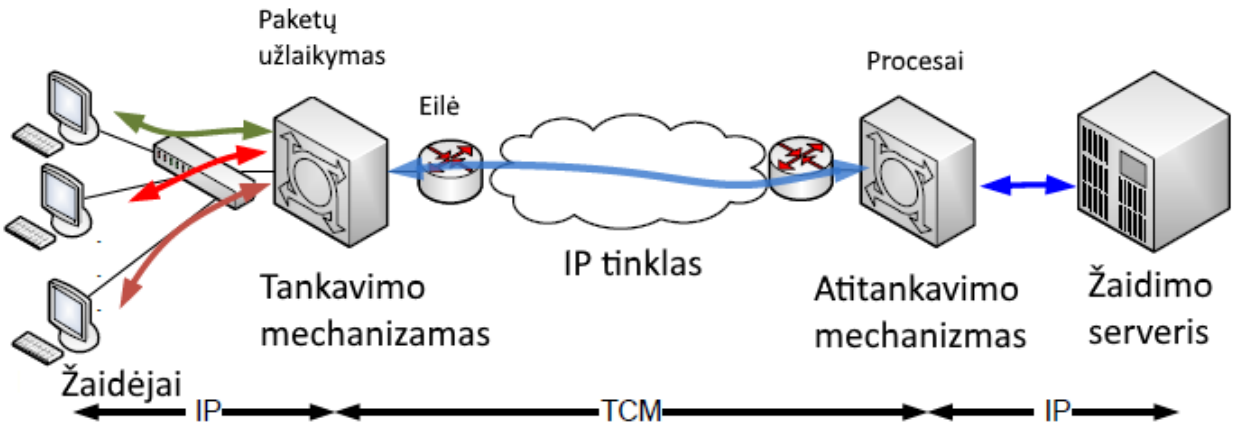
TCM (tuneliavimas, suspaudimas ir tankinimas) [SSN+12] tai tokia duomenų srauto optimizavimo technologija, kurios panaudojimas internetiniuose MMORPG žaidimuose pirmą kartą pristatytas 2012 metais, kompiuterių ir telekomunikacinių sistemų konferencijoje, Italijoje. Ji suteikia galimybę sparčiai perduoti TCP duomenų paketus. Norint optimizuoti duomenų srautą ir padidinti siuntimo spartą TCM technologija galima pritaikyti tokiuose scenarijuose, kur tam tikras skaičius žaidėjų dalinasi tuo pačiu duomenų perdavimo kanalu, pavyzdžiui tarp įgaliotojo serverio (angl. *proxy server*) ir žaidimo serverio, kaip pavaizduota 11 pav.



11 pav. TCM technologijos naudojimo scenarijus

Įgaliotasis serveris gauna duomenų srautą iš žaidėjų esančių tam tikroje zonoje, dažniausiai miesto ar rajono, ir tuomet persiunčia šį srautą į pagrindinį žaidimo serverį. Taip surinktas duomenų srautas įgaliotojo serverio pusėje gali būti suspaustas, naudojant duomenų paketų antraščių suspaudimo algoritmus, po to sutankintas į vieną duomenų perdavimo kanalą naudojant PPPMUX tankintuvą [TKW05]. Tuomet pasitelkus PPP ir L2TPv3 tuneliavimo technologija [LTG05], galima perduoti suspaustą duomenų srautą gavėjui, šiuo atveju, žaidimo serveriui.

Šios technologijos efektyvumas atsiskleidžia, kai sutankintas duomenų srautas yra perduodamas iš įgalioto į žaidimo serverį vienu duomenų perdavimo kanalu. Taip yra pasidalinama bendra antraštė tarp visų suspaustų duomenų paketų, tačiau antraštės dydis šiek tiek padidėja dėl tuneliavimo technologijos. Kaip matome 12 pav., tunelis yra sukuriamas tarp tankintuvų esančių įgalioto ir žaidimo serverio, kur po to paketai yra atkuriami lygiai tokie patys, kokie buvo išsiųsti žaidimo klientų. Taip pat tankinimą galime naudoti ir atvirkštiniu būdu - duomenų srauto perdavimui iš žaidimo serverio į klientų žaidimus, persiunčiant juos per įgaliotąjį serverį.

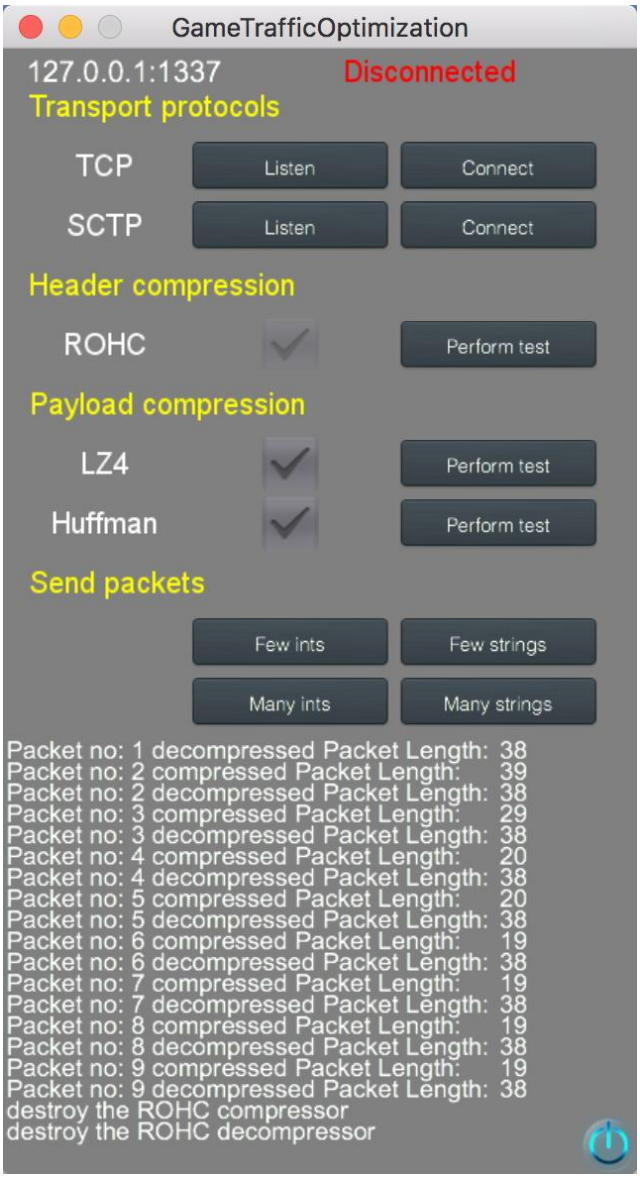


12 pav. TCM technologijos schema su visais joje susijusiais elementais

Straipsnyje [SSN+12] yra rašoma, kad pasitelkus internetinio žaidimo „World of Warcraft“ duomenų srauto modeliu [SKR07], sudaryto iš realių duomenų ir „Matlab“ programos pagalba buvo sukurta simuliacija, kuri naudojo TCM technologiją. Iš šios simuliacijos surinkti duomenys parodė, kad duomenų srauto optimizaciją tokiame scenarijuje gali būti optimizuojama iki 60 procentų. Tokią technologiją galima būtų taikyti ir interneto kavinėse (angl. *Internet café*).

5. Tiriamų technologijų realizacija

Norint įvertinti technologijų naudą duomenų srauto optimizacijoms buvo būtina jas realizuoti taip, kad patogiai leistų atlikti aktualius temai tyrimus. Dėl šios priežasties buvo sukurtas įrankis, matomas 13 pav. Įrankis turi valdymo skydą, kuriame galima atlikti tyrimus. Tyrimai buvo aprašomi tolimesniuose tiriamų technologijų skyriuose.



13 pav. Autoriaus sukurtas realizuotų technologijų valdymo skydas

Įrankis buvo sukurtas veikti ant įvairių platformų: Windows, Mac OS, Linux, iOS, Android. Tai suteikė galimybę įvertinti tiriamų technologijų rezultatus skirtingoms kompiuterinėms

architektūroms. Taip pat įrankis gali funkcionuoti kaip tariamas žaidimo serveris arba klientas, klausytis ateinančių duomenų paketų ir siųsti juos. Šį funkcionalumą galime įjungti paspaudus „Listen“ ir „Connect“ mygtukus, kurie atitinkamai nustato ar įrankis turės klausytis naujai ateinančių lizdų ar bandyti jungtis prie serverio, nurodant IP adresą ir kanalą (angl. *Port*). Įjungus porą ar daugiau šio įrankio egzempliorių atsiranda galimybė užmegzti ryšį tarp jų. Sėkmingai sudarius ryšį tarp serverio ir kliento(-ų) atsiranda galimybė perduoti duomenų paketus nuspaudžiant atitinkamus mygtukus esančius žemiau užrašo „Send packets“. Užmegzti ryšį yra svarbu tam, kad būtų galima stebėti ir įvertinti duomenų paketų perdavimą iš vieno įrankio egzemplioriaus į kitą. Sekančių technologijų – ROHC, LZ4, Huffman jau parengtus tyrimus galima įgyvendinti paspaudus atitinkamus, šalia technologijų pavadinimų, esančius mygtukus. Taip pat, valdymo skyde uždėjus varneles, galime pritaikyti tos technologijos funkcionalumus vėliau siunčiamiems duomenų paketams. Tarpiniai rezultatai bus išvedami apatinėje įrankio dalyje, kur yra palikta daugiau vietos. Kitiems rezultatams surinkti ir įvertinti bus naudojami specialūs įrankiai aprašomi sekančiame skyriuje.

5.1. Tyrimuose naudojami įrankiai

Dalis rezultatų buvo surenkami pasinaudojant sukurtu įrankiu – tiriamų technologijų valdymo skydu, tačiau kita dalis rezultatų buvo surenkama pasinaudojant kitais įrankiais todėl, kad šie įrankiai sugeba patogiau ir detaliau aptikti norimus rezultatus.

Buvo naudoti šie įrankiai:

- Wireshark – populiariausias internetinių protokolų analizavimo įrankis. Sugeba aptikti įvairius protokolus, detaliai juos išanalizuoti bei pateikti grafikus.
- Cocoa Packet Analizer – programa leidžia analizuoti Mac operacinės sistemos surenkamus paketus.

6. Duomenų paketų antraščių optimizacijos tyrimai

Šiame skyriuje yra aprašomi transporto protokolų ir duomenų paketų antraščių suspaudimo technologijų tyrimai, kurie leidžia įvertinti kiekvienos pasirinktos architektūros skirtumus, jų privalumus bei trūkumus. Taip pat aprašomi tyrimo rezultatai bei jų surinkimo būdai. Yra siekiama įvertinti pasirinktų technologijų galimybę optimizuoti duomenų srautą internetiniams žaidimams.

6.1. Transporto protokolų ir duomenų paketų antraščių suspaudimo tyrimų aprašymas

Tyrimų tikslas yra surasti transporto protokolą, kuris neprarandant duomenų, perduotų duomenų paketus su mažiausia antrašte kas paveiktų didesnę paketų perdavimo spartą ir sumažintų duomenų srautą. Taip pat surinkti rezultatus apie duomenų paketų antraščių dydžius bei juos ištirti. Tokiems tyrimams atlikti buvo realizuotas duomenų perdavimas TCP ir SCTP transporto protokolais.

Dėl didesnio žaidėjų skaičiaus ties internetiniais asmens vaidinimo žaidimais ir paketų perdavimo užtikrinimo nebuvimo, buvo nuspręsta atsisakyti tirti UDP transporto protokolą. UDP protokolą dažniausiai naudoja internetiniai pirmo asmens šaudyklės žaidimai, kuriems patikimumas nėra svarbus, tačiau šis žanras užima nedidelę dalį ties įvairių internetinių žaidimų žanrų.

Taip pat buvo realizuota ROHC technologija, kuri geba suspausti duomenų paketų antraštes. Tačiau toks antraščių suspaudimas įmanomas tik su TCP, UDP, ir RTP transporto protokolais. Šiais tyrimais yra siekiama surasti optimalų sprendimą ar sprendimų kombinaciją, kuri padėtų optimizuoti duomenų srautą internetiniams žaidimams.

Van Jacobson duomenų paketų antraščių suspaudimo algoritmo nuspręsta nenagrinėti, nes taikant šį suspaudimo algoritmą ant senų serijinių ir telefoninio ryšio (angl. *dialup*) modemų buvo pastebėta [Jac90], kad iškyla nemažai klaidų, nes duomenų paketo apkrovos integralumo tikrinimo mechanizmas remiasi kontrolinės sumos laukais, o šios technologijos atspaudimo mechanizmas galėdavo atkurti ir pažeistus paketus, kai dalis duomenų paketų apkrovos dėl blogo ryšio dingdavo, o atspaudimo mechanizmas atkurdavo tariamai korektišką kontrolinę sumą. Dėl tokios priežasties mes negalime taikyti šios technologijos internetiniuose žaidimuose, nes jiems yra būtina užtikrinti ar visi duomenis bus teisingi.

6.2. Tyrimų sąlygos

Norint rasti tinkamus sprendimus mažinti duomenų srautui internetiniams žaidimams yra įvardijamos tokios sąlygos:

- Tiriama technologija neturi paveikti paketų perdavimo. Visi siunčiami duomenų paketai turi pasiekti gavėją;
- Iš visų tiriamų technologijų duomenų paketų antraščių dydis ir perdavimo sparta turi būti labiausiai optimali. Technologijos turi sumažinti paketų antraščių dydžius ir taip paketas pasiektų gavėją greičiau.

6.3. Duomenų paketų antraščių dydžio rezultatų surinkimas

Norint išmatuoti duomenų paketų antraščių dydžius reikėjo užmegzti ryšį tarp duomenų siuntėjo ir gavėjo ir ištirti duomenų srautą. Tai buvo įgyvendinta pasinaudojant sukurtuoju valdymo skydu. Įjungus porą jo egzempliorių ir nurodžius atitinkamus IP adresus bei kanalus buvo užmegztas ryšis. Pasinaudojus sukurtuoju įrankiu yra išsiunčiami duomenų paketai su jau paruoštomis apkrovomis. Pasinaudojus Wireshark įrankiu tokie duomenų paketai buvo sugaunami. Wireshark įrankiui sugavus perduodamus paketus, jame galima matyti visus transporto protokolo nustatymus bei duomenų paketų turinį. ROHC technologijos tyrimų duomenims, kurie yra aprašyti žemiau, surinkti buvo naudojamas valdymo skydo rezultatų išvedimo langas.

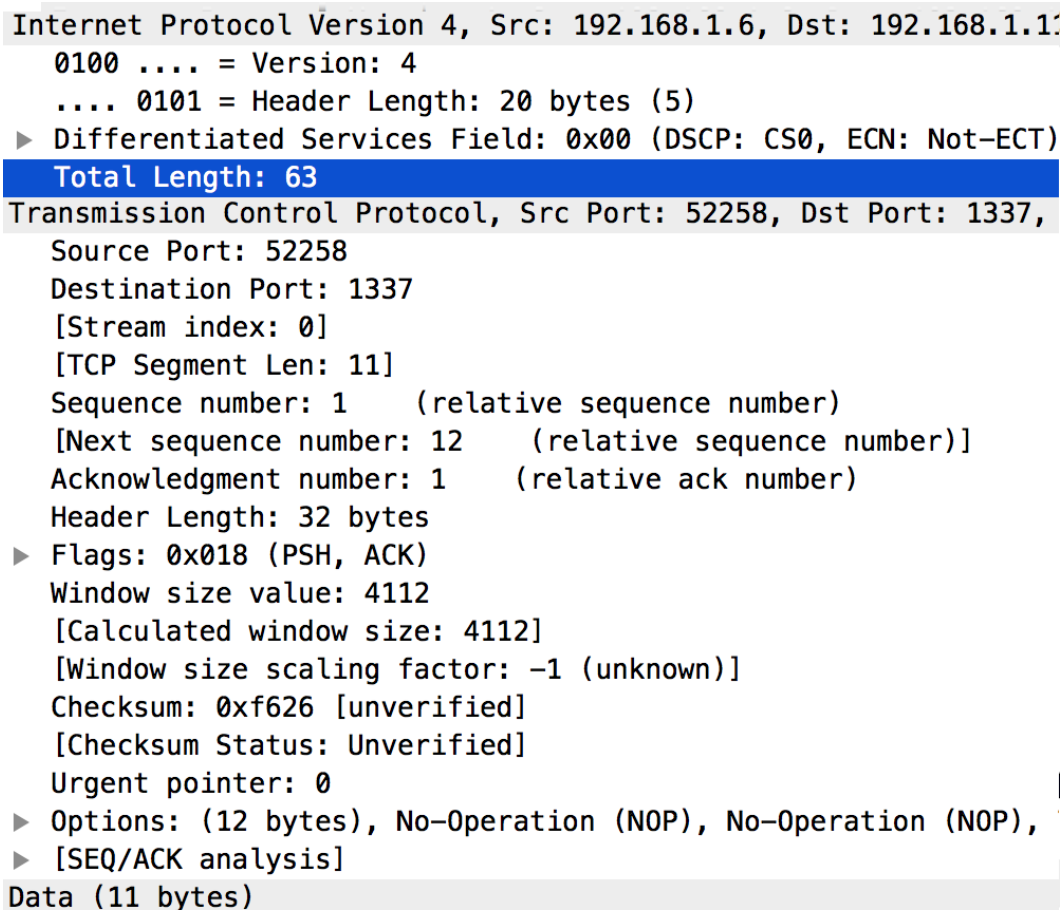
6.4. TCP transporto protokolo tyrimas

TCP – labiausiai paplitęs protokolas, ypatingai MMORPG žanro žaidimuose, kuris užtikrina, kad visi perduodami duomenų paketai pasieks žaidimo klientą ir jie pasieks jį jiems numatyta eilės tvarka. Tačiau tokie privalumai prideda daugiau duomenų siunčiant paketus, o perduodamų duomenų dydis daro įtaką paketų perdavimo spartai – kuo didesnis duomenų paketas, tuo visas jo turinys bus siunčiamas ilgiau.

Šiame tyrime buvo ištirtas šio protokolo antraštės dydis su įjungtais TCP parametrais, kurie yra naudojami internetiniuose žaidimuose:

- Išjungtas Naglio algoritmas [Nag84], kuris paveikia duomenų perdavimo spartą, laukiant 500 milisekundžių papildomų duomenų paketų, kai apkrova yra ganėtinai maža (65,535 baitai) – TCP lango skalės (angl. *TCP window scale*) dydis [Pos81] tam, kad kelios apkrovos būtų sujungtos į vieną duomenų paketą.
- Išjungtas blokavimo režimas (Šaltinis: <https://www.scottklement.com/rpg/socktut/nonblocking.html>). TCP protokole yra numatytas parametras, kuris įgalina blokavimo režimą. Kai yra bandoma skaityti iš srauto ar yra atvykusių duomenų, atsakymas yra grąžinamas, kai yra nuskaitomas nors vienas baitas duomenų arba praeina tam tikras laiko tarpas (500 milisekundžių). Šis laukimo procesas vadinamas „blokavimu“. Tas pats galioja ir su rašymų į srautą operacija – kol visi duomenys nebus surašyti į srautą, ryšys yra „užblokuojamas“. Toks parametras dažniausiai yra išjungiamas internetiniuose žaidimuose, nes juose neturi būti papildomo uždelsimo atliekant įvairius žaidimo veiksmus.
- Įjungtas binarinio formato duomenų perdavimo režimas. Toks perdavimas yra tipinis internetiniams žaidimams. Kiekvieno duomenų paketo pradžioje prideda 2 baitus duomenų, kurie nurodo, kokio dydžio yra perduodama duomenų paketo apkrova.

Tyrimo metu buvo perduodamas duomenų paketas tarp valdymo skydo egzempliorių su 9 baitų apkrova. Kaip matome 14 pav. Wireshark įrankis sugavo duomenų paketą, kurio dydis 63 baitai – 20 baitų IP antraštė, 32 baitai TCP antraštė, kurios 12 baitų sudaro TCP protokolo nustatymai. Toliau seka 11 baitų apkrovos - 2 baitai apkrovos ilgis, 9 baitai patys perduodami duomenys.



14 pav. Wireshark įrankio sugauto TCP/IP duomenų paketo turinys

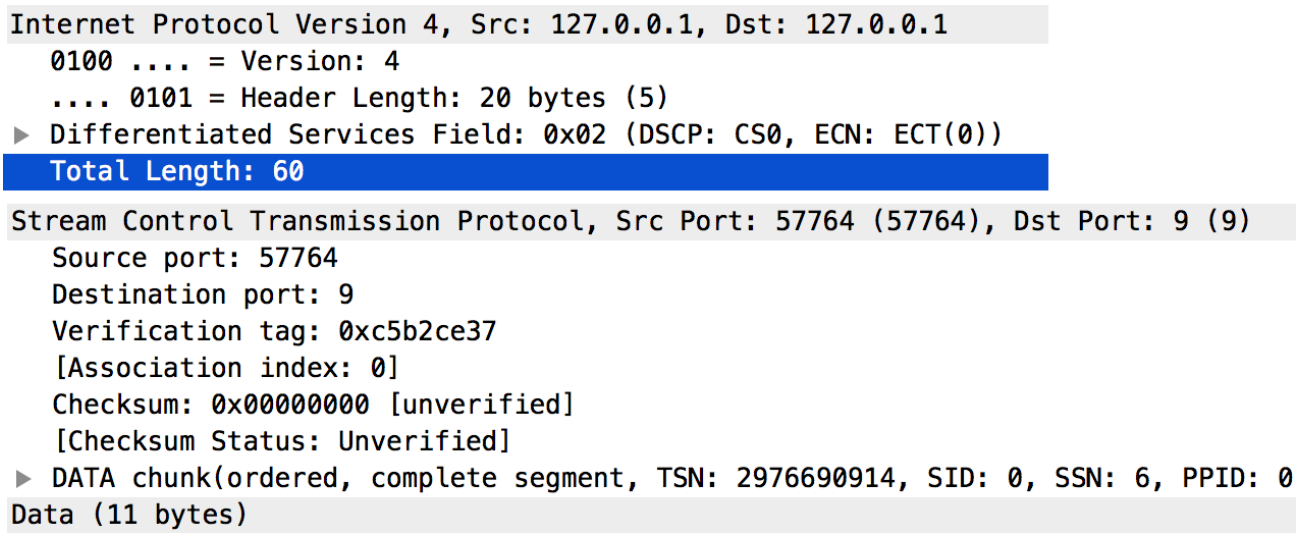
Rezultatai parodė TCP antraštės dydį su įjungtais TCP parametrais, kurie yra naudojami internetiniuose žaidimuose – 32 baitai. Šis dydis bus naudingas vertinant sekančio transporto protokolo antraštės dydį.

6.5. SCTP transporto protokolo tyrimas

SCTP transporto protokolas [AIN+06] turi kelias savybes, kurios būtų naudingos internetinių žaidimų tinkle ir kurių neturi kiti transporto protokolai. Tačiau tokios savybės prideda SCTP duomenų paketui papildomas dalis, kurios didina pačio paketo dydį. Taip pat protokolo standartiniai nustatymai sukelia didelę duomenų perdavimo delsą, todėl straipsnyje yra rekomenduojama šio protokolo nenaudoti internetiniuose žaidimuose [Hed11]. Tačiau yra taip pat rašoma, kad modifikuojant SCTP protokolo parametrus galima būtų pasiekti geresnius duomenų perdavimo spartos rezultatus už TCP protokolo [PBP+09].

Šiame tyrime buvo ištirtas SCTP duomenų paketo antraštės dydis modifikuojant šio protokolo parametrus, bei įgalinant įvairias jo savybes, kurios būtų naudingos internetiniams žaidimams.

Visų pirma buvo ištirtas SCTP duomenų paketas su numatytais parametrais. Kaip matome 15 pav. Wireshark įrankis sugavo SCTP duomenų paketą, kurio visas dydis 60 baitų – 20 baitų IP antraštė, 11 baitų apkrova, o likę 29 baitai SCTP dalis (angl. *chunk*).



15 pav. Wireshark įrankio sugauto SCTP duomenų paketo turinys

Šiame tyrime buvo parinkti ir įgalinti SCTP parametrai [STP+11], kurie numatomai būtų reikalingi internetiniams žaidimams:

- Išjungta SCTP delsa (angl. *SCTP no delay*) – įjungtas parametras panašiai veikia kaip jau aprašytas TCP Naglio algoritmas. Laukiant tam tikrą laiko tarpą sujungia kelis mažus duomenų paketus į vieną ir kartu juos išsiunčia. Toks funkcionalumas buvo išjungiamas naudojant TCP transporto protokolą, todėl taip pat jį reikia išjungti ir SCTP transporto protokole;
- Sumažintas retransliacijos delsos laikas RTO (angl. *retransmission timeout*). Šis parametras nusako, kiek transporto protokolui reikia laukti laiko, kad būtų išsiųstas pakartotinis duomenų paketas, kai prieš tai siųstas nepasiekė gavėjo (laukia ilgai patvirtinimo iš gavėjo ar negavo siųsto duomenų paketo). SCTP tokio parametro laikas yra numatytas 60 sekundžių. Toks laikas būtų ženkliai didelis internetiniams žaidimams, kai pradingus žaidėjo siųstam duomenų paketui jo žaidimo klientas sustotų atlikinėti kitus veiksmus ir lauktų 60 sekundžių atsako. Tas pats buvo minėta analitinėje dalyje, straipsnyje [Hed11] vertino šio protokolo realizaciją internetiniams žaidimams, tačiau ši mintis buvo atmesta dėl netinkamų šio protokolo numatytų

parametrų. Tačiau kaip ir straipsnyje [PBP+09] rašo, tinkamai modifikavus šiuos parametrus būtų galima pasiekti geresnius už TCP protokolo paketų perdavimo spartos rezultatus. Tai įsivertinus buvo pakeistas šio parametro delsos laiką į 1 sekundę. Taip pat atsižvelgiant į tai, kad TCP protokolo retransliacijos delsos laikui yra numatytos 3 sekundes (šis laikas kinta TCP transporto protokole per specifinius šio protokolo algoritmus [PAC+11]);

- Įjungtas spartaus perleidimo (angl. *fast handoff*) parametras. Duomenims keliaujant internetiniu srautu jie yra perduodami per įvairius maršrutizatorius. Toks perdavimas yra vadinamas perleidimu (angl. *handover*). Per tokį perleidimą yra laiko periodas, per kurį duomenis priimantysis negali siųsti ar gauti duomenų paketų, dėl ryšio perjungimo uždelsimo ir IP protokolo operacijų. Šis laiko periodas yra vadinamas perleidimo delsa (angl. *handover latency*) [Koo08]. Šis parametrais įjungia spartaus perleidimo SCTP algoritmus, kurie sumažina IP protokolo operacijų skaičių.

Žemiau, 16 pav., matome SCTP protokolo su įvardintais parametrais perduoto duomenų paketo turinį. Visas duomenų paketo dydis 56 baitai – iš jų 5 baitų apkrova, 20 baitų IP antraštė ir likę 31 baitai SCTP dalis.

```
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
   0100 .... = Version: 4
   .... 0101 = Header Length: 20 bytes (5)
▶  Differentiated Services Field: 0x02 (DSCP: CS0, ECN: ECT(0))
   Total Length: 56
Stream Control Transmission Protocol, Src Port: 53531 (53531), Dst Port: 9 (9)
   Source port: 53531
   Destination port: 9
   Verification tag: 0x66d490d8
   [Association index: 0]
   Checksum: 0x00000000 [unverified]
   [Checksum Status: Unverified]
▶  DATA chunk(ordered, complete segment, TSN: 3504482130, SID: 0, SSN: 1, PPID: 0
Data (5 bytes)
```

16 pav. Wireshark įrankio sugauto SCTP su parametrais duomenų paketo turinys

SCTP transporto protokolas turi pakankamai daug parametrų, kur dalis jų būtų naudinga internetiniams žaidimams. Šiame tyrime buvo įgalinti aktualiausi parametrai, kurie numatomai dalinai išspręstų straipsnyje [Hed11] įvardijamas problemas.

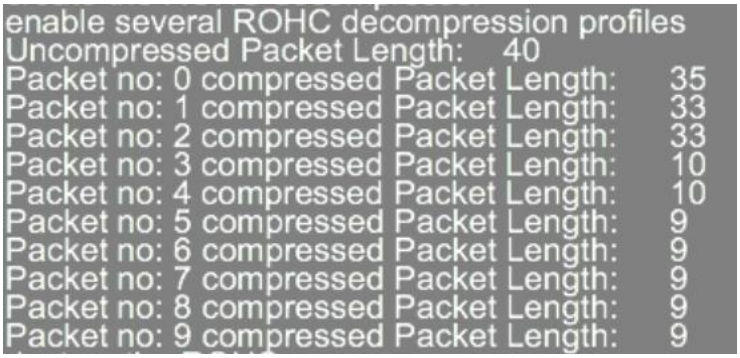
6.6. ROHC technologijos tyrimas

ROHC yra tvirtas antraščių suspaudimo karkasas (angl. *robust header compression*) [Bor01], skirtas suspausti IP, UDP, RTP ir TCP internetinių paketų antraštes. ROHC suglaudina tipinę TCP/IP 40 baitų antraštę iki kelių baitų, priskiriant suspaudimo ir išskleidimo mechanizmą, prieš kanalą ir kanalo gale. Suspaudimo mechanizmas suspaudžia antraštę iki kelių baitų, o atspaudimo mechanizmas iš tų kelių baitų atkuria pilną antraštę.

Tyrimo metu buvo ištirtas TCP IP protokolo antraštės suspaudimo laipsnis ir perduodamų apkrovos dydžio įtaka jam. Tai yra ar apkrovos dydis daro įtaką antraštės suspaudimo laipsniui. Taip pat reikėjo ištirti ar atspaudžiamas duomenų paketas yra pilnai atkuriamas, ar apkrovos duomenis neišsikraipo.

Tyrimams atlikti buvo paruošti du TCP/IP duomenų paketai, vienas iš jų buvo be apkrovos, o kitas turėjo apkrovą, kurios dydis buvo 50 baitų. Atitinkamai viso duomenų paketo dydžiai buvo 40 ir 90 baitų. Buvo atliekami du bandymai, kuriuose buvo paleidžiamas ciklas, kuriame eilės tvarka buvo vykdomi tokie veiksmai: duomenų paketo suspaudimas, suspausto dydžio išvedimas, duomenų paketo atspaudimas, atspausto dydžio išvedimas. Pirmame bandyme buvo spaudžiami ir renkami rezultatai iš duomenų paketo be apkrovos, antrame – su apkrova.

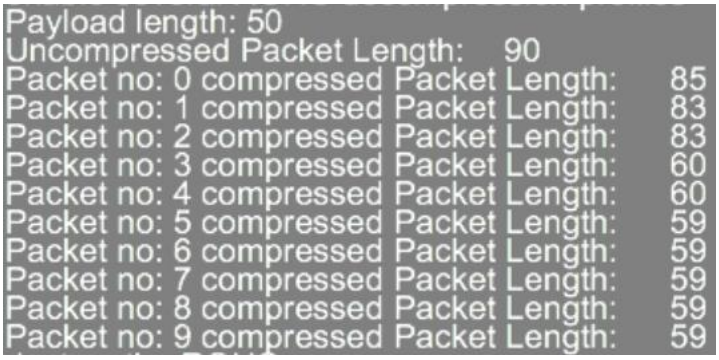
Kaip matome 17 pav., pirmųjų 5 suspaustų duomenų paketų dydžiai mažėjančiai kinta, o tolimesnių duomenų paketų dydis įgavo pastovų sumažintą dydį – 9 baitus. Pasinaudojus ROHC technologija 40 baitų duomenų paketas (IP antraštė – 20 baitų, TCP – 20 baitų) buvo suspaustas iki 9 baitų.



17 pav. ROHC technologijos pirmo bandymo rezultatai

Žemiau, 18 pav., matome antro bandymo rezultatus – pirmųjų 5 duomenų paketų dydžiai mažėjančiai kinta, o tolimesnių duomenų paketų dydis įgavo pastovų sumažintą dydį – 59 baitus.

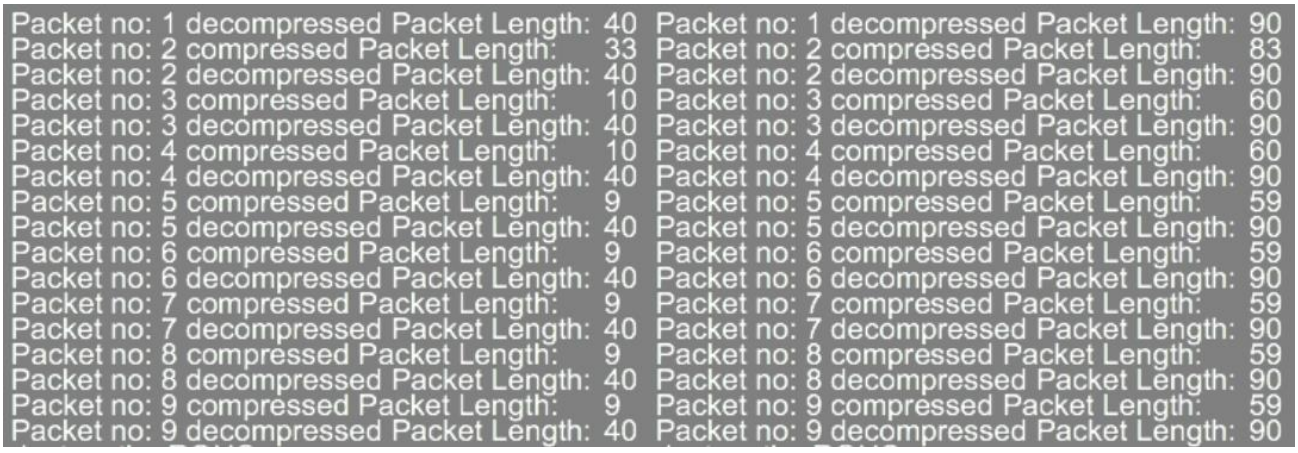
Pasinaudojus ROHC technologija 90 baitų duomenų paketas (IP antraštė – 20 baitų, TCP – 20 baitų, apkrova – 50) buvo suspaustas iki 59 baitų (9 baitai suspaustoji antraštė, o kiti 50 – apkrova). ROHC technologija apkrovos nesuspaudžia, todėl ji lieka tokio pat dydžio.



18 pav. ROHC technologijos antro bandymo rezultatai

Pirmųjų 5 duomenų paketų dydis yra didesnis nei likusių suspaustų paketų. Toks reiškinys yra normalus. Tai nutinka todėl, kad suspaudimo mechanizmas pirmuose paketuose prideda papildomų duomenų, reikalingų atspaudimo mechanizmui, kurie padėtų atkurti duomenų paketą į originalų jo dydį. Šie bandymai parodė, kad apkrovos dydis nepaveikia suspaudimo laipsnio.

Žemiau, 19 pav., matome atspaudimo rezultatus. Kairėje rezultatų pusėje matomas pirmo bandymo kiekvienas atspausto duomenų paketų dydis, o dešinėje pusėje – antro bandymo atspaudimo rezultatai.



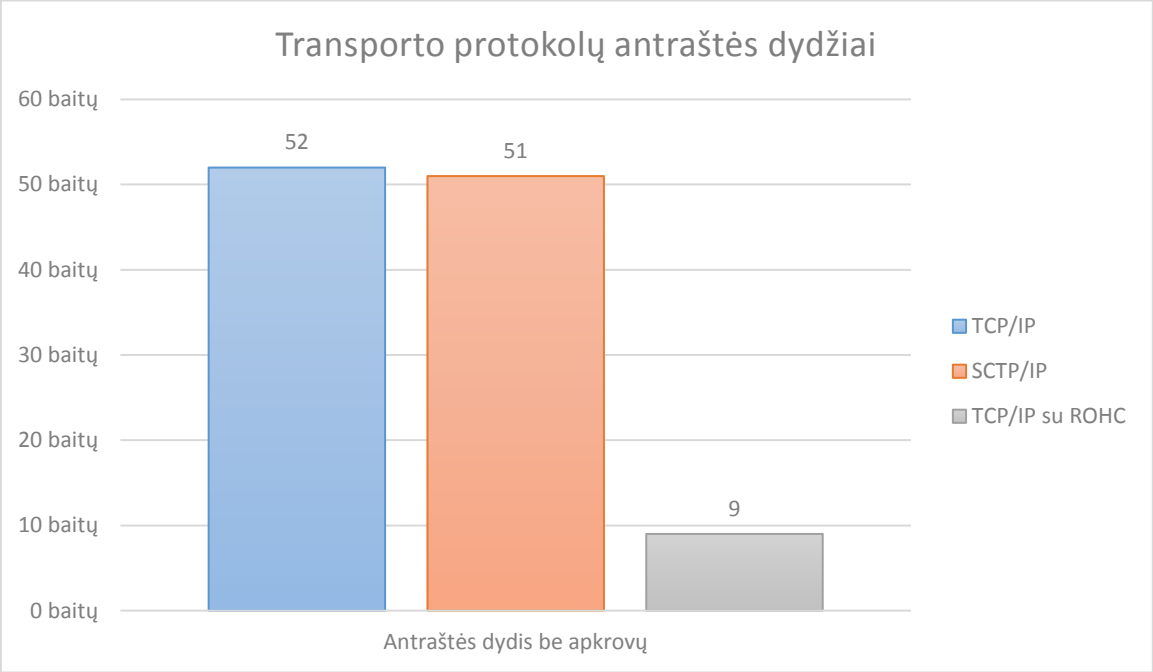
19 pav. ROHC bandymų atspaudimo rezultatai

Šie rezultatai parodo, kad suspaudime dalyvaujantys apkrovos duomenys neišsikraipo, tai yra atspaudžiant yra gaunami tie patys originalūs duomenys.

6.7. Apibendrinimas

Šiame skyriuje buvo aprašyti transporto protokolų bei duomenų paketų antraščių suspaudimo tyrimai. Buvo išanalizuoti ir ištirti TCP ir SCTP transporto protokolai. Iš tyrimų buvo surinkti antraščių dydžiai su nustatytais žaidimams reikalingais parametrais. Taip pat ištirta ROHC antraštės suspaudimo technologija. Nustatytas TCP/IP protokolo antraštės suspaudimo laipsnis ir perduodamų apkrovos dydžio įtaka jam.

Rezultatai, matomi 20 pav., parodė, kad SCTP antraštės dydis yra mažesnis už TCP vienu baitu, tačiau pritaikius ROHC antraštės suspaudimo technologiją, TCP paketo antraštė žymiai sumažėja - 40 baitų TCP/IP antraštę suspaudžiama iki 9 baitų. ROHC technologijos nėra galimybės pritaikyti SCTP transporto protokolui, todėl TCP gerokai daugiau sutaupo duomenų srauto perduodant duomenų paketus.



20 pav. Transporto protokolų antraščių dydžiai

7. Duomenų paketų apkrovos optimizacijos tyrimai

Šiame skyriuje yra aprašomi duomenų paketų apkrovos suspaudimo technologijų tyrimai, kurie leidžia įvertinti kiekvienos pasirinktos architektūros skirtumus, jų privalumus bei trūkumus. Taip pat aprašomi tyrimo rezultatai bei jų surinkimo būdai. Yra siekiama įvertinti pasirinktų technologijų galimybę optimizuoti duomenų srautą internetiniams žaidimams.

7.1. Duomenų paketų apkrovos suspaudimų tyrimų aprašymas

Analitinėje dalyje buvo daryta prielaida, kad tiriamų algoritmų duomenų paketų apkrovos suspaudimo laipsnis gali būti priklausomas nuo perduodamų duomenų kiekio ir tipo. Dėl šios priežasties reikia įvertinti tipinius duomenis, kurie yra perduodami internetiniuose žaidimuose. Taip pat reikia apskaičiuoti algoritmų suspaudimo bei atspaudimo spartą ant skirtingų kompiuterinių architektūrų, norint geriau suvokti suspaudimo laipsnio reikšmingumą.

7.2. Tyrimų sąlygos

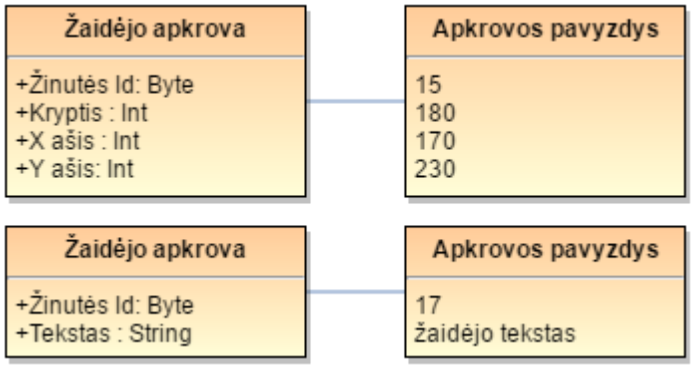
Norint rasti tinkamus sprendimus mažinti duomenų paketų apkrovas yra įvardijamos tokios sąlygos:

- Tiriamų suspaudimo algoritmų skaičiavimo laikas neturi didinti žaidimo gaišties laiko;
- Jei suspaudimo algoritmai perduoda papildomus atkodavimui reikalingus duomenis (tokius kaip Huffman medis) tai visas suspausto duomenų paketo dydis neturi viršyti nesuspausto paketo dydžio.

7.3. Internetinių žaidimų duomenų paketų tipai ir jų turinys

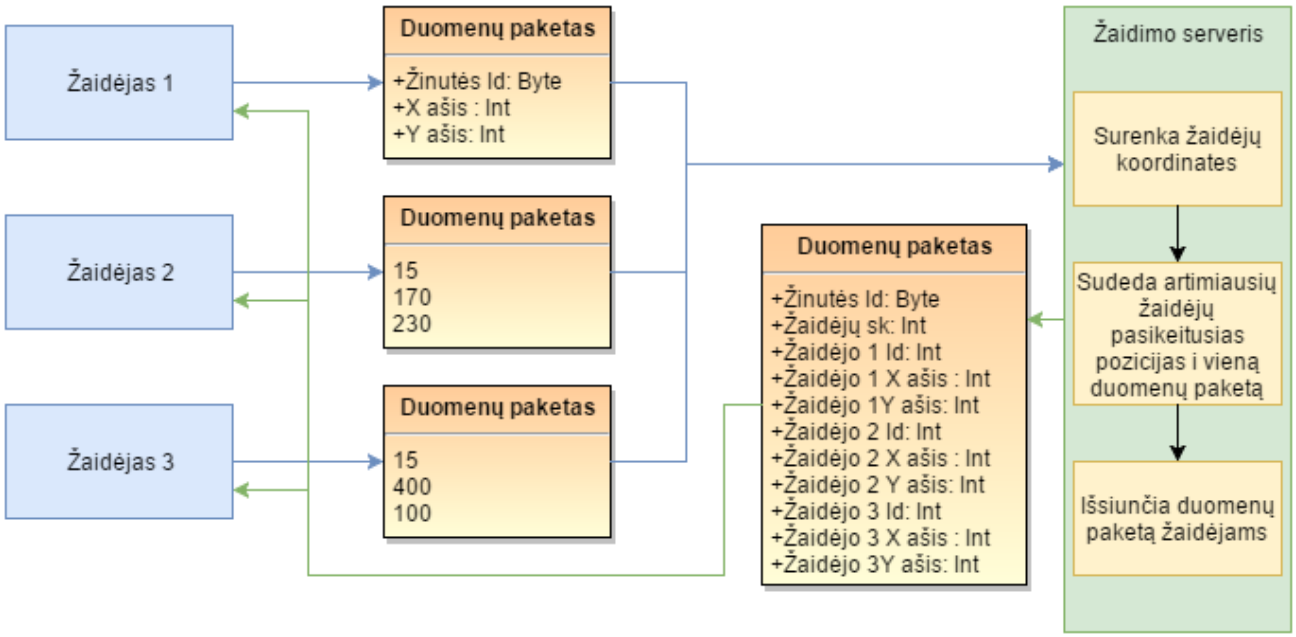
Norint ištirti duomenų paketų apkrovos suspaudimo galimybes, reikia įvardyti pagrindinius žaidimuose perduodamų duomenų tipus bei jų turinį. Iš išanalizuotų duomenų srauto modelių internetiniuose žaidimuose [CHH+15] yra nustatyta, kad daugiausia duomenų paketų yra

siunčiama dėl žaidėjų pozicijų atnaujinimo. Kaip matome 21 pav. į tokias duomenų paketų apkrovas dažniausiai patenka apkrovos tipo identifikacinis laukas, kuris skirtas atpažinti, kokius duomenis siunčia žaidėjas, po to seka personažo koordinačių bei krypties duomenys. Kitas dažnas duomenų paketų apkrovos tipas yra žaidėjų susirašinėjimo žinutės, į kurias patenka apkrovos tipo identifikacinis laukas bei žaidėjo siunčiamas tekstas.



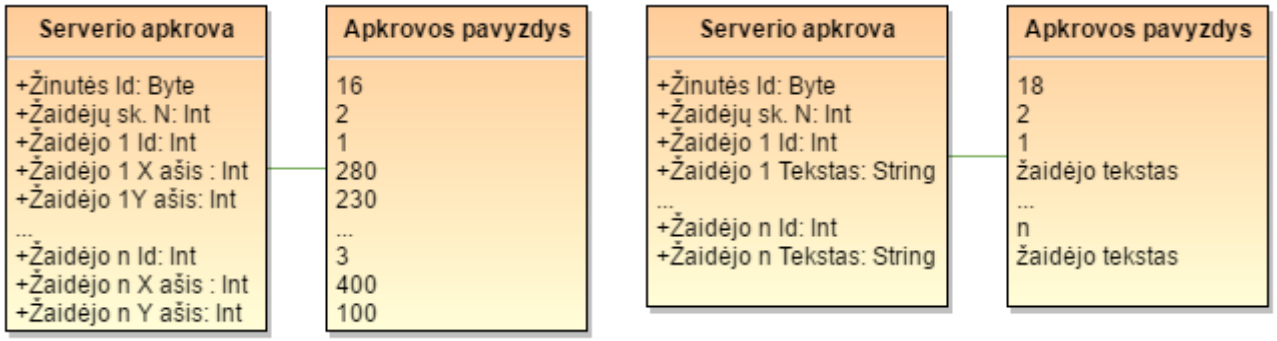
21 pav. Tipinės duomenų paketų apkrovos internetiniuose žaidimuose

Internetiniuose žaidimuose, kuriuose personažas yra valdomas kompiuterio pele, personažo koordinatės yra išsiunčiamos iš kliento į žaidimo serverį po pelytes paspaudimo, o žaidimuose, kuriuose personažas yra valdomas klaviatūros klavišais, duomenys yra perduodami pastoviai tam tikru intervalu, dažniausiai kas sekundę. Kaip matome 22 pav. iš serverio pusės tokio tipo duomenys per tam tikrą intervalą, dažniausiai kas sekundę, yra surenkami, sugrupuojami pagal žaidimo zonas. Taip yra sukuriamas didesnis duomenų paketas, kuriame sudėtos visos, tam tikros zonos, žaidėjų pozicijos ir toks paketas yra išsiunčiamas visiems toje zonoje esantiems žaidėjams. Tokiu pat būdu yra sugrupuojamos serverio pusėje žaidėjų susirašinėjimo žinutės.



22 pav. Tipinių duomenų paketų apkrovų perdavimas internetiniuose žaidimuose

Bandant suspausti duomenų paketus žaidėjų pusėje yra tikimybė, kad tokie duomenų paketai tik išaugs, o ne sumažės, nes perduodamų kiekis yra ganėtinai mažas, o tam tikri suspaudimo algoritmai į suspaustus duomenis kartu prideda papildomą informaciją, skirtą atspaudimo mechanizmui sėkmingai atspausti juos. Taip turėtų nutikti naudojant Huffman suspaudimo algoritmą, nes jame turi būti perduotas ir Huffman medis. Darant tokias prielaidas labiau yra orientuojama į didesnį duomenų kiekį, matomą 23 pav. Tai yra į duomenis, kuriuos išsiunčia žaidimo serveris. Taip pat su šia prielaida ir žinant, kad algoritmų suspaudimo bei atspaudimo greitis nėra vienodas galima manyti, kad žaidimo serveriui yra reikalingas suspaudimo algoritmas, kurio yra didesnė suspaudimo sparta, o žaidimo klientui yra reikalingas algoritmas, kuris turi didesnę atspaudimo spartą.



23 pav. Tipinės duomenų paketų apkrovos siunčiamos iš žaidimo serverio

7.4. Žaidimo apkrovų suspaudimo laipsnio rezultatų surinkimas

Tyrime buvo surenkami rezultatai parodantys internetinių žaidimų tipinių duomenų paketų apkrovų suspaudimo laipsnį. Tyrimams atlikti reikėjo imituoti internetiniuose žaidimuose perduodamus srautu duomenis.

Tyrimas buvo atliktas pasinaudojus sukurtuoju valdymo skydu ir sudarant TCP ryšį tarp jo egzempliorių. Juo buvo perduodami skirtingi duomenys:

- mažas kiekis skaitmeninių duomenų, kas atitinka tipinį žaidėjo pozicijų atnaujinimo duomenų paketą, siunčiamą iš žaidimo kliento;
- didelis kiekis skaitmeninių duomenų, kas atitinka tipinį žaidėjo pozicijų atnaujinimo duomenų paketą, siunčiamą iš žaidimo kliento;
- mažas kiekis tekstinių duomenų, kas atitinka tipinį žaidėjo susirašinėjimo žinučių duomenų paketą, siunčiamą iš žaidimo kliento;
- didelis kiekis tekstinių duomenų, kas atitinka tipinį žaidėjo susirašinėjimo žinučių duomenų paketą, siunčiamą iš žaidimo serverio.

Perduodami duomenų paketai buvo ištirti ir rezultatai išvedami į valdymo skydo rezultatų lauką.

7.5. Suspaudimo algoritmų skaičiavimo laiko rezultatų surinkimas

Tyrime buvo surenkami rezultatai parodantys algoritmų suspaudimų bei atspaudimų skaičiavimo laiko spartą, išreikštą megabaitais per sekundę mato vienetu (MB/s). Duomenys buvo surinkti pasinaudojus sukurtuoju valdymo skydu, kuriuo buvo suspaudžiamas didelis duomenų kiekis – 100 MB (100’000’000 baitų). Šis duomenų kiekis yra sugeneruojamas testo pradžioje su atsitiktinai parengtomis „Char“ duomenų tipo reikšmėmis – atsitiktiniais skaičiais ir raidėmis. Suspaudimo pradžioje ir pasibaigus skaičiavimui yra fiksuojamas laikas. Tuomet kai žinome per kiek laiko yra suspaudžiama 100MB išvedame proporciją ir apskaičiuojame kiek megabaitų geba algoritmas suspausti per sekundę. Rezultatas yra išvedamas į valdymo skydo rezultatų vietą. Šį veiksmą kartojame kelis kartus, kad būtų galima išvesti duomenų vidurkį. Kadangi kiekvieną kartą

yra sugeneruojami skirtingi duomenys skaičiavimo laikas taip pat skiriasi su kiekvienu kartu. Toliau seka tokia pat eilės tvarka atspaudimo spartos skaičiavimas. Šie tyrimai buvo atliekami ant kelių skirtingų kompiuterinių platformų.

7.6. Huffman technologijos tyrimas

Huffman suspaudimo algoritmas [Huf07] yra paremtas simbolių dažnumu duomenyse, kuriuos norime suspausti. Pritaikant Huffman algoritmą, dažniausiems simboliams yra suteikiama mažiau bitų, o retesniems daugiau. Taip yra sukuriamas Huffman medis, kuris yra naudojamas duomenų atspaudimui. Manoma, kad Huffmano suspaudimo algoritmas skaitmeniniams, žaidėjų pozicijų atnaujinimų duomenimis, nebūtų idealus, nes simbolių dažnumas žaidėju koordinatėse nėra žinomas iš karto dar neturint tų koordinačių, todėl Huffman medis turėtų būti siunčiamas kartu su koordinatėmis. Medžio dydis gali atsverti suspaustų duomenų dydį. Tačiau šį suspaudimą galima būtų taikyti paketuose su didesniu duomenų skaičiumi.

Tyrimo metu buvo ištirtas duomenų paketo apkrovos suspaudimo laipsnis pasinaudojus Huffman suspaudimo algoritmą. Taip pat skirtingų duomenų tipų įtaka suspaudimo laipsniui. Tai yra ar skaitiniai, tekstiniai duomenys bei jų kiekis daro įtaką apkrovos suspaudimo laipsniui. Taip pat reikėjo ištirti algoritmo skaičiavimo laiko spartą, ant kelių skirtingų kompiuterinių architektūrų.

Tyrimams atlikti buvo paruošti 4 TCP/IP duomenų paketai, su skirtingomis apkrovomis. Kiekvienai iš apkrovų tipui yra suteikti atitinkami pavadinimai:

- „Few ints“ – skaitmeniniai duomenys, 20 baitų;
- „Many ints“ – skaitmeniniai duomenys, 200 baitų;
- „Few strings“ – tekstiniai duomenys, 50 baitų;
- „Many strings“ – tekstiniai duomenys, 500 baitų.

Buvo atlikti du tyrimai, viename fiksuojamas suspaudimo laipsnis kiekvienai iš minėtų duomenų paketų apkrovai. Kitame buvo skaičiuojama algoritmo suspaudimo bei atspaudimo sparta. Antras tyrimas buvo pakartotinai atliekamas ant 3 skirtingų kompiuterinių architektūrų.

Rezultatai yra pateikti 3 ir 4 lentelėse:

3 lentelė. Huffman algoritmo suspaudimo laipsnio rezultatai

Skaičiuojama reikšmė	**„Few ints“**	**„Few strings“**	**„Many ints“**	**„Many strings“**
Suspaudimo laipsnis	-60%	-204%	32%	72,8%

4 lentelė. Huffman algoritmo skaičiavimo laiko rezultatai

Kompiuterinės platformos specifikacija	**Suspaudimo sparta**	**Atspaudimo sparta**
Windows OS, CPU: 64-bits, Intel Core i7-3632QM, 2,2 GHz	143,53 MB/s	59,83 MB/s
Mac OS, CPU: dual-core Intel Core m7 64-bit, 3,7 GHz	180,31 MB/s	105,58 MB/s
iOS (iPhone 5), CPU: dual-core 32-bit ARMv7-A, 1,3 GHz	84,00 MB/s	53,83 MB/s

Kaip ir buvo daryta prielaida, Huffman algoritmas, mažam duomenų kiekiui, duomenų paketo apkrovai prideda papildomo dydžio. Tai yra perduodamas Huffman medis atsveria perduodamų duomenų kiekį. Taip pat matome, kad didesnius tekstinius duomenis šis algoritmas suspaudžia geriau nei skaitmeninius. Be to algoritmas suspaudžia duomenis daug greičiau nei juos atspaudžia.

7.7. LZ4 technologijos tyrimas

Lempel-Ziv yra universalus nenuostolingas glaudinimo algoritmas. Pristatytas Terry Welch, kaip patobulinimas jau egzistuojančio LZ78 algoritmo [Wel84]. Duomenys yra suslegiami ieškant vienodų sekų, vadinamų frazėmis. Rastos sekos yra išsaugomos lentelėje ir joms yra skiriami trumpesni 8 bitų raktai.

Tyrimo metu buvo ištirtas duomenų paketo apkrovos suspaudimo laipsnis pasinaudojus LZ4 suspaudimo algoritmą. Taip pat skirtingų duomenų tipų įtaka suspaudimo laipsniui. Tai yra ar skaitiniai, tekstiniai duomenys bei jų kiekis daro įtaką apkrovos suspaudimo laipsniui. Taip pat reikėjo ištirti skaičiavimo laiko spartą ant kelių skirtingų kompiuterinių architektūrų.

Tyrimams atlikti buvo paruošti 4 TCP/IP duomenų paketai, su skirtingomis apkrovomis jau minėtomis praeitame skyriuje. Buvo atlikti du tyrimai, viename fiksuojamas suspaudimo laipsnis kiekvienai iš minėtų duomenų paketų apkrovai. Kitame buvo skaičiuojama algoritmo suspaudimo bei

atspaudimo sparta. Antras tyrimas buvo pakartotinai atliekamas ant 3 skirtingų kompiuterinių architektūrų.

Rezultatai yra pateikti 5 ir 6 lentelėse:

5 lentelė. LZ4 algoritmo suspaudimo laipsnio rezultatai

Skaičiuojama reikšmė	**„Few ints“**	**„Few strings“**	**„Many ints“**	**„Many strings“**
Suspaudimo laipsnis	-10 %	-36%	78%	86,3%

6 lentelė. LZ4 algoritmo skaičiavimo laiko rezultatai

Kompiuterinės platformos specifikacija	**Suspaudimo sparta**	**Atspaudimo sparta**
Windows OS, CPU: 64-bits, Intel Core i7-3632QM, 2,2 GHz	112,97 MB/s	134,05 MB/s
Mac OS, CPU: dual-core Intel Core m7 64-bit, 3,7 GHz	173,03 MB/s	179,94 MB/s
iOS (iPhone 5), CPU: dual-core 32-bit ARMv7-A, 1,3 GHz	75,88 MB/s	80,99 MB/s

Iš rezultatų matome, kad mažo kiekio duomenis algoritmas suspaudžia neefektyviai, prideda papildomo dydžio. Taip pat matome, kad skaitmeninius duomenis šis algoritmas suspaudžia geriau nei tekstinius. Be to algoritmas atspaudžia duomenis daug greičiau nei juos suspaudžia.

7.8. SIMD-FastPFOR technologijos tyrimas

FastPFOR [VS14] algoritmas siekia kaip įmanoma dažniau išnaudoti SIMD komandas [DLN15] skaičių masyvo suspaudimui pasitelkus bitų maskavimo technologija. Toks skaičių masyvo suspaudimas būtų naudingas suspaudžiant duomenų paketų apkrovas turinčias žaidėjų pozicijų atnaujinimo duomenis, kadangi būtų perduodamos žaidėjų koordinačių skaitmenų sąrašas.

Tyrimo metu buvo ištirtas duomenų paketo apkrovos suspaudimo laipsnis pasinaudojus SIMD-FastPFOR suspaudimo algoritmą. Buvo spaudžiami tik skaitmeniniai duomenys nes algoritmas tik jiems yra pritaikytas. Taip pat reikėjo ištirti skaičiavimo laiko spartą ant kelių skirtingų kompiuterinių architektūrų.

Tyrimams atlikti buvo paruošti 2 TCP/IP duomenų paketai, su skirtingomis apkrovomis jau minėtomis praeitame skyriuje. Buvo atlikti du tyrimai, viename fiksuojamas suspaudimo laipsnis kiekvienai iš minėtų duomenų paketų apkrovai. Kitame buvo skaičiuojama algoritmo suspaudimo bei atspaudimo sparta. Antras tyrimas buvo pakartotinai atliekamas ant kelių skirtingų kompiuterinių architektūrų.

Rezultatai yra pateikti 7 ir 8 lentelėse:

7 lentelė. SIMD-FastPFOR algoritmo suspaudimo laipsnio rezultatai

Skaičiuojama reikšmė	**„Few ints“**	**„Few strings“**	**„Many ints“**	**„Many strings“**
Suspaudimo laipsnis	20 %	-	48%	-

8 lentelė. SIMD-FastPFOR algoritmo skaičiavimo laiko rezultatai

Kompiuterinės platformos specifikacija	**Suspaudimo sparta**	**Atspaudimo sparta**
Windows OS, CPU: 64-bits, Intel Core i7-3632QM, 2,2 GHz	213,82 MB/s	323,90 MB/s
Mac OS, CPU: dual-core Intel Core m7 64-bit, 3,7 GHz	250,23 MB/s	349,12 MB/s
iOS (iPhone 5), CPU: dual-core 32-bit ARMv7-A, 1,3 GHz	-	-

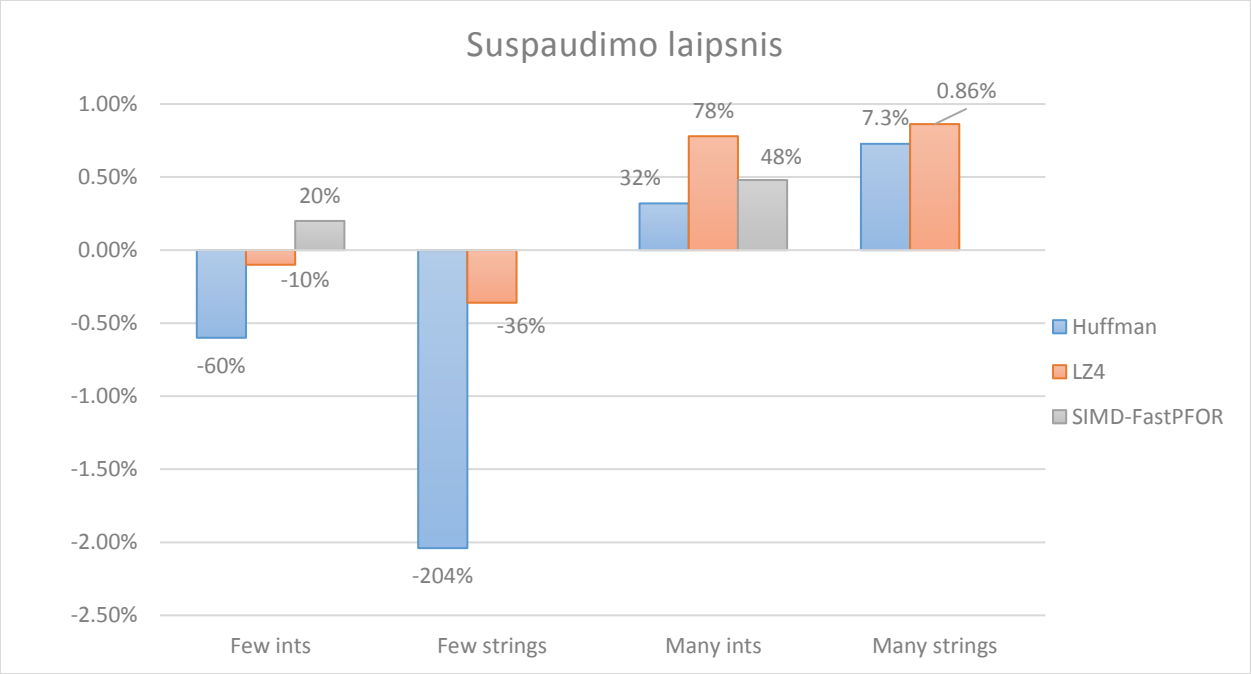
Iš rezultatų matome, kad net mažo kiekio skaitmeninius duomenis algoritmas pajėgia suspausti. Taip pat algoritmas atspaudžia ir atspaudžia duomenis daug greičiau nei Huffman ir LZ4 algoritmai.

7.9. Apibendrinimas

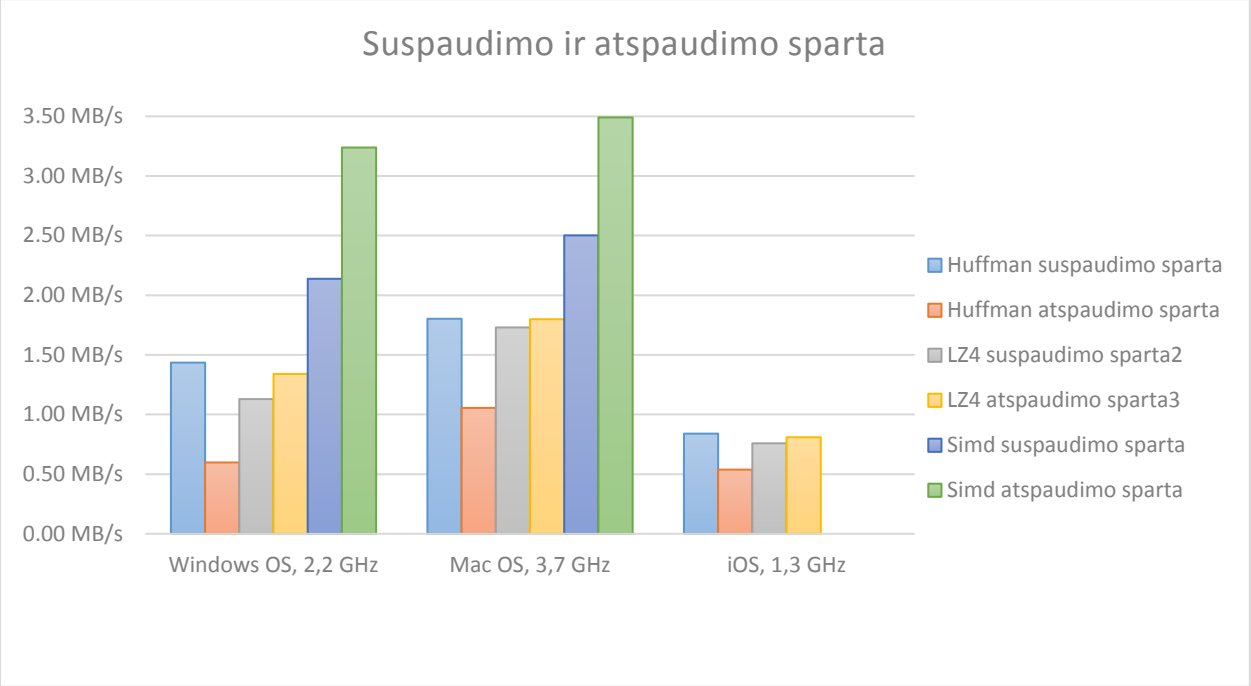
Šiame skyrelyje buvo aprašyti suspaudimo algoritmų tyrimai. Aprašyti internetinių žaidimų pagrindiniai duomenų paketų tipai ir apkrovų turiniai. Tyrimuose buvo įvertintas skirtingų duomenų paketų apkrovų suspaudimo laipsnis bei skaičiavimo laiko sparta.

Buvo daryta prielaida, kad žaidimo serveriui yra reikalingas suspaudimo algoritmas, kurio yra didesnė suspaudimo sparta ir suspaudimo laipsnis. Tokiu atveju SIMD-FastPFor algoritmas yra tinkamiausias, tačiau algoritmas suspaudžia tik skaitmeninius duomenis. Tekstinius duomenis suspausti tiktu naudoti LZ4 suspaudimo algoritmą.

Tyrimų rezultatai, matomi 24 pav., parodė, kad SIMD-FastPFor sugeba suspausti itin mažus duomenis, tačiau suspaudžia tik skaitmeninius duomenis. Huffman algoritmas pasižymėjo šiek tiek didesne suspaudimo sparta nei LZ4, tačiau atspaudimo sparta ir suspaudimo laipsnis yra ženkliai mažesnis. Huffman ir LZ4 algoritmai nesugeba suspausti itin mažų duomenų, nes yra perduodama papildoma informacija skirta atspaudimo mechanizmams duomenims atspausti, kuri pranoksta perduodamų duomenų kiekį. Taip pat rezultatai, matomi 25 pav., parodė, kad greičiausiai skaitmeninius duomenis suspaudžia SIMD-FastPFor algoritmas.



24 pav. Algoritmų suspaudimo laipsnių rezultatai



25 pav. Algoritmų suspaudimo ir atspaudimo spartos rezultatai

8. Duomenų paketų perdavimo dažnio optimizacijos tyrimai

Šiame skyriuje yra aprašomi duomenų paketų perdavimo dažnio mažinimo metodų pritaikymai internetiniams žaidimams. Yra pateikiami metodų taikymo scenarijai, kurie leidžia įvertinti jų privalumus bei trūkumus. Yra siekiama įvertinti pasirinktų metodų galimybę optimizuoti duomenų srautą internetiniams žaidimams.

8.1. Duomenų paketų perdavimo dažnio mažinimo tyrimų aprašymas

Surasti sprendimus, kurie nemažintų žaidėjo pasitenkinimo žaidimu, tačiau padėtų mažinti duomenų srautą specifiniuose scenarijuose. Pateikti scenarijaus pavyzdžių.

8.2. Tyrimų sąlygos

Norint rasti tinkamus metodus yra įvardijamos tokios sąlygos:

- Metodas neturi sumažinti žaidėjų pasitenkinimo žaidimu;
- Metodas turi mažinti duomenų paketų dažnį.

8.3. Žaidėjų išskaidymas į mažesnes grupes

Žaidėjų išskaidymas į apribotas virtualias erdves, dar žinomoms kaip „Instance dungeon“ [SDM09], sumažina duomenų perdavimo skaičių tarp greta esančių žaidėjų. Priklausomai nuo žaidimo, žaidėjai, norintys gauti specifinių artefaktų ar kitų privalumų, kurių paprastai nėra galimybės gauti, privalo išsiskirstyti į mažas grupes, dažniausiai 5-10 žaidėjų, taip gaunant galimybę patekti į tam tikras virtualaus pasaulio vietas. Kelios grupės žaidėjų gali patekti į tas pačias vietas, tačiau tos grupės viena kitos nematys, visi veiksmai apribotose vietose yra atliekami paraleliai. Yra potenciali galimybė taikyti šį metodą be didelių apribojimų, kad žaidėjai galėtų mėgautis žaidimu nepraleidžiant daug laiko tokiose erdvėse.

Pavyzdys kaip sumažėtų perduodami duomenys tarp greta esančių žaidėjų: žaidime yra virtualus pasaulis, kuriame yra šimtas žaidėjų, tai vietoj to, kad bet kuris vienas žaidėjas pakeisdamas savo poziciją perduotų savo pozicijos duomenis kitiems 99 žaidėjams, tai sukuriame du virtualius

pasaulius, ir taip yra tikėtina, kad dalis žaidėjų pasirinks vieną kelią, o kita dalis kitą kelią, taip nebereikės vieno žaidėjo duomenų perduoti visiems žaidėjams, o tik daliai jų.

8.4. Neaktyviems žaidėjams perduoti mažiau duomenų

Aptinkant neveiksnius žaidėjus, kurių avatarai ilgą laiką stovi vienoje vietoje ir neatlieka jokių veiksmų, jiems būtų galima siųsti mažiau duomenų paketų, užslepiant aktyvius žaidėjus, arba tam tikrus jų veiksmus, iki tol kol neveiksnus žaidėjas sugrįš atgal į žaidimą ir taps veiksniu.

Iš [CHL05] straipsnyje pateiktų statistinių duomenų yra žinoma, kad „ShenZhou Online“ internetiniame žaidime daugiausia žaidėjų yra neveiksnūs, o straipsnyje [VM09] yra teigiama, kad MMORPG žaidimuose žaidėjai būna neveiksnūs pusę viso savo sugaišto žaidime laiko. Taip pat teigiama, kad žaidėjai būna dažniausiai neveiksnūs didžiausiuose virtualaus pasaulio miestuose, kur žaidėjų skaičius yra didžiausias. Tokiu atveju, žaidėjų užslėpimas neveiksniems žaidėjams atneštų ganėtinai reikšmingą naudą optimizuojant duomenų srautą.

8.5. Apibendrinimas

Šiame skyriuje buvo pateikti dveji internetinių žaidimų scenarijų pavyzdžiai, kuriuose aptariami metodai, galintys sumažinti perduodamų duomenų paketų dažnį, nedarant įtakos žaidėjų pasitenkinimui žaidimu.

REZULTATAI

Aptarti duomenų srauto optimizavimo būdai. Išanalizuoti transporto protokolai:

- UDP – dažniausiai naudojamas FPS žanro internetiniuose žaidimuose, nes tokiems žaidimams nėra būtina užtikrinti ar visi perduodami duomenų paketai pasieks žaidimo klientą. Šiuo protokolu duomenys yra perduodami greičiau, nes nėra tikrinamas duomenų paketų eiliškumas taip pat prarasti paketai nėra pakartotinai išsiunčiami.
- TCP – labiausiai paplitęs protokolas, ypatingai MMORPG žanro žaidimuose, kuris užtikrina, kad visi perduodami duomenų paketai pasieks žaidimo klientą ir jie pasieks jį numatyta eilės tvarka. Tačiau tokie privalumai prideda daugiau duomenų siunčiant paketus, o perduodamų duomenų dydis įtakoja paketų perdavimo spartą – kuo didesnis duomenų paketas, tuo visas jo turinys bus siunčiamas ilgiau. Yra galimybė UDP protokolą naudoti kartu su TCP – vienu protokolu būtų perduodami duomenys reikalaujantys mažesnės delsos, tačiau nereikalaujantys pristatymo užtikrinimo, o kitu protokolu - priešingo konteksto duomenys.
- SCTP – panašus į TCP protokolą, orientuotas į patikimą duomenų perdavimą buvo sukurtas stengiantis pašalinti TCP protokolo trūkumus. Užtikrina apsaugą nuo pasiskirstytų atsisakymo aptarnauti atakų (DDoS). Šis protokolas yra ganėtinai naujas lyginant su kitais protokolais, todėl nėra plačiai naudojamas. Turi papildomų savybių kaip kelių dažnių ir kanalų išskyrimas, kurios būtų naudingos internetinių žaidimų tinkluose.

Taip pat aptarti šie duomenų paketų suspaudimo algoritmai:

- Van Jacobson algoritmas– tai TCP/IP duomenų paketų antraščių suspaudimo technologija, skirta didinti duomenų paketų atsako laiką mažos spartos serijiniuose modemuose. Tipinę 40 baitų TCP/IP duomenų paketų antraštę sugeba suspausti net iki 5 baitų, išimant pasikartojančius laukus, bei suteikiant pasikeitimo deltą laipsnį laukams kurie keičiasi.
- ROHC – tai įvairių transporto protokolų antraščių suspaudimo karkasas. Geba suspausi tipinę 40 baitų TCP/IP antraštę į kelis baitus, naudojant mažiausiai reikšmingo bito algoritmą LSB. Naudojant šį algoritmą, reiktų perduoti ir duomenų kontekstą, ko nereikalauja Van Jacobson suspaudimo algoritmas, tačiau šiuo karkasu galima būtų suspausti ne tik TCP antraštes, tačiau ir UDP.
- LZW – tai universalus nenuostolingas glaudinimo algoritmas, plačiai naudojamas įvairių programinių įrangų. Analizuojant buvo svarstoma šio algoritmo pritaikymas suspaudžiant duomenų paketų apkrovą. Straipsniuose aprašyti rezultatai parodė, kad suspaudimo laipsnis

didėja kai perduodami duomenys saugo daug tekstinių duomenų, tačiau reikalauja daug skaičiavimo laiko. Yra potencialios galimybės taikyti šį suspaudimo būdą perduodant tekstinius duomenis internetiniuose žaidimuose, tokius kaip susirašinėjimai tarp žaidėjų ir perspėjimo žinutės žaidime.

- Huffman – tai yra nenuostolingas glaudinimo algoritmas. Straipsniuose aprašyti rezultatai parodė didesnį našumą, nei LZW, lyginant suspaudimo laipsnio ir skaičiavimo laiko santykį. Yra potencialios galimybės taikyti šį suspaudimo būdą perduodant skaitmeninius duomenis, tokius kaip žaidėjų pozicijų atnaujinimai ir žaidimo kovų statistika.
- SIMD - komandų naudojimas leidžia žymiai pagreitinti slankų kablelį naudojančių programų greitį, nes viena procesoriaus komanda lygiagrečiai yra apdorojami keli duomenų elementai.

Taip pat išanalizuoti metodai, kurie mažina duomenų paketų perdavimo dažnį:

- Apribotos virtualios erdvės – žaidėjai yra išskaidomi grupėmis, nematant viena kitos, kas mažina perduodamus duomenis tarp žaidėjų. Tačiau metodas dažnai sukelia žaidėju nepasitenkinimą žaidimu, nes žaidėjai būna priversti ieškoti komandos narių, kurie norėtų visi eiti į vieną iš apribotų virtualių erdvių. Yra potenciali galimybė taikyti šį metodą be didelių apribojimų, kad žaidėjai galėtų mėgautis žaidimu nepraleidžiant daug laiko tokiuose erdvėse.
- Platūs virtualieji pasauliai – kuo labiau žaidėjai išsiskaido po visą virtualų pasaulį, tuo labiau siunčiamų duomenų paketų skaičius mažėja, nes žaidėjai atitolsta vienas nuo kito ir serveris neprivalo siųsti žaidėjo veiksmų tiems žaidėjams, kurie jo nemato. Tačiau virtualiuose pasauliuose dažniausiai būna pagrindinės susitikimų vietos, tokios kaip pagrindiniai žaidimo miestai, kuriuose greta esančių žaidėjų skaičius išlieka ganėtinai aukštas. Jei šis metodas būtų tinkamai suderintas su tam tikromis žaidimo funkcijomis, tai yra potenciali galimybė sumažinti žaidėjų skaičių didelėse susitikimo vietose.
- Žaidimo kanalų išskyrimas - šis metodas išskaido žaidėjų skaičių grupėmis, kurios viena kitos nemato visą žaidimo sesijos laiką. Šio metodo trūkumas yra toks, kad priverčia žaidėjus patiems išsiskirstyti per visus žaidimo kanalus. Jei žaidėjas norės žaisti kartu su draugais, visi jie turės iš karto susitarti kuriame kanale žais. Tačiau ši technologija praverstu nenumatytuose scenarijuose, kai žaidėjų skaičius neplanuotai padidėtų ir reiktų skubios išeities gebėti aptarnauti visus žaidėjus.
- Žaidėjų užslėpimas – šis metodas pirmą kartą buvo pastebėtas 2016 metais, viename MMORPG žaidime. Jis buvo netinkamai realizuotas ir kėlė žaidėjų nepasitenkinimą žaidimu.

Tačiau buvo aptartas būdas kaip tinkamai realizavus šį metodą galima būtų gauti ganėtinai reikšmingą naudą optimizuojant duomenų srautą.

Aptartas tuneliavimo ir tankavimo technologijų realizavimas duomenų srauto optimizacijai internetiniuose žaidimuose:

- TCM – šią technologiją yra siekiama sukaupti dalį duomenų srauto įgaliotojo serverio pusėje, visų duomenų paketų antraštes suspausti, po to sutankuoti į vieną duomenų paketą ir pasinaudojus tuneliavimo technologija perduoti šį suspaustą dalį srauto žaidimo serveriui. Specifiniame scenarijuje galima būtų optimizuoti iki 60 procentų duomenų srauto taikant šią technologiją. Tokią technologiją galima būtų taikyti ir interneto kavinėse.

Realizuotas duomenų paketų perdavimas skirtingais transporto protokolais, kuriais pasinaudojus buvo galima atlikti aktualius temai transporto protokolų tyrimus:

- TCP – realizavus šį transporto protokolą buvo ištirtas šio protokolo antraštės dydis, su įjungtais TCP parametrais, kurie yra naudojami internetiniuose žaidimuose;
- SCTP – realizavus šį transporto protokolą buvo ištirtas SCTP duomenų paketo antraštės dydis modifikuojant šio protokolo parametrus, bei įgalinant įvairias jo savybes, kurios būtų naudingos internetiniams žaidimams.

Realizuota duomenų paketų antraščių suspaudimo technologija, dėl kurios buvo atliktas duomenų paketų antraščių suspaudimo tyrimas:

- ROHC – realizavus šią duomenų paketų antraščių suspaudimo technologiją buvo ištirtas TCP/IP protokolo antraštės suspaudimo laipsnis ir perduodamos apkrovos dydžio įtaka jam.

Realizuotos duomenų paketų apkrovos suspaudimo technologijos, kuriomis pasinaudojant buvo atlikti duomenų paketų apkrovos suspaudimo tyrimai:

- Huffman – realizavus šią duomenų suspaudimo technologiją buvo ištirtas duomenų paketo apkrovos suspaudimo laipsnis, skirtingų duomenų tipų įtaka jam, bei skaičiavimo laikas skirtinguose kompiuterinėse architektūrose;
- LZ4 – realizavus šią duomenų suspaudimo technologiją buvo ištirtas duomenų paketo apkrovos suspaudimo laipsnis, skirtingų duomenų tipų įtaka jam, bei skaičiavimo laikas skirtinguose kompiuterinėse architektūrose;

- SIMD-FastPFOR – realizavus šią duomenų suspaudimo technologiją buvo ištirtas duomenų paketo apkrovos suspaudimo laipsnis, skirtingų duomenų tipų įtaka jam, bei skaičiavimo laikas skirtinguose kompiuterinėse architektūrose.

Ištirti duomenų paketų perdavimo dažnio mažinimo metodai, pateikti internetinių žaidimų scenarijų pavyzdžiai, kuriuose būtų galima sumažinti duomenų paketų perdavimo dažnį, nedarant įtakos žaidėjų pasitenkinimui žaidimu:

- Žaidėjus išskaidyti į mažesnes grupes. Priklausomai nuo žaidimo, žaidėjai norintys gauti specifinių artefaktų ar kitų privalumų, kurių paprastai nėra galimybės gauti, privalo išsiskirstyti į mažas grupes, dažniausiai 5-10 žaidėjų, taip gaunant galimybę patekti į tam tikras virtualaus pasaulio vietas;
- Perduoti mažesnį duomenų kiekį neaktyviems žaidėjams. Aptinkant neveiksnius žaidėjus, kurių avatarai ilgą laiką stovi vienoje vietoje ir neatlieka jokių veiksmų, jiems būtų galima siųsti mažiau duomenų paketų, užslepiant aktyvius žaidėjus, arba tam tikrus jų veiksmus iki tol, kol neveiksnus žaidėjas sugrįš atgal į žaidimą ir taps veiksniu.

IŠVADOS

Tyrimuose buvo surinkti rezultatai iš kurių buvo vertinama technologijų nauda duomenų srauto optimizacijai internetiniams žaidimams:

- Palyginti TCP ir SCTP transporto protokolų antraščių dydžiai. Rezultatai parodė, kad SCTP antraštės dydis yra mažesnis už SCTP vienu baitu, tačiau pritaikius ROHC antraštės suspaudimo technologiją, TCP paketo antraštė žymiai sumažėja - 40 baitų TCP/IP antraštę suspaudžiama iki 9 baitų. ROHC technologijos nėra galimybės pritaikyti SCTP transporto protokolui, todėl TCP gerokai daugiau sutaupo duomenų srauto perduodant duomenų paketus;
- Atrinktas efektyviausias duomenų paketo suspaudimo algoritmas transporto protokolams. Rezultatai parodė, kad SIMD-FastPFor algoritmas yra greičiausias ir sugeba suspausti itin mažus duomenis, tačiau suspaudžia tik skaitmeninius duomenis. Huffman algoritmas pasižymėjo šiek tiek didesne suspaudimo sparta nei LZ4, tačiau atspaudimo sparta ir suspaudimo laipsnis yra ženkliai mažesnis. Huffman ir LZ4 algoritmai nesugeba suspausti itin mažų duomenų, nes yra perduodama papildoma informacija skirta atspaudimo mechanizmams duomenims atspausti, kuri pranoksta perduodamų duomenų kiekį.
- Atrinkti virtualaus pasaulio suskaidymo metodai, kurie sumažina duomenų paketų dažnį, neigiamai neįtakojant pasitenkinimo žaidimu. Taip pat sukurtos rekomendacijos surastiems sprendimams taikyti internetiniuose žaidimuose.

Atlikta rezultatų analizė, iš kurios buvo įvardinta tinkamiausia, sprendimų kombinacija, kuri padėtų mažinti duomenų srautą internetiniams žaidimams:

- Naudoti TCP transporto protokolą kartu su ROHC antraštės suspaudimo technologija;
- Suspausti iš serverio siunčiamų duomenų paketų apkrovas pasinaudojant SIMD-FastPFor ir LZ4 suspaudimo algoritmais;
- Perduoti mažesnį duomenų kiekį neaktyviems žaidėjams;
- Žaidėjus išskaidyti į mažesnes grupes.

ŠALTINIAI

[AIN+06] Amer, P. Iyengar, J. Natarajan, P. & Stewart, R., SCTP: An Innovative Transport Layer Protocol for The Web, Singapore, 2006. Prieiga per internetą: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.146.2166>

[ASM+11] B. Anand, J. Sebastian, Soh Yu Ming, A. L. Ananda, Mun Choon Chan, and R. K. Balan, PGTP: Power Aware Game Transport Protocol for Multi-Player Mobile Games, Singapore, 2011.

[Bor01] C. Bormann, "RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed." RFC 3095, IETF Network Working Group, Liepa 2001.

[Bra13] R. Braden, Requirements for Internet Hosts - Communication Layers. RFC 1122 (Orginalas), Spalis 1989. Atnaujintos RFC versijos 1349, 4379, 5884, 6093, 6298, 6633, 6864 – Vasaris, 2013.

[CHH+15] Kuan-Ta Chen, Chun-Ying Huang, P. Huang, Chin-Laung Lei, An Empirical Evaluation of TCP Performance in Online Games, Institute of Information Science, Taiwan, 2015.
[žiūrėta 2015-12-12]. Prieiga per internetą:
<http://www.iis.sinica.edu.tw/~swc/pub/tcp_in_games.html>

[CHL05] Kuan-Ta Chen, Polly Huang, and Chin-Laung Lei, Game Traffic Analysis: An MMORPG Perspective, Institute of Information Science, Taiwan, 2005.
[žiūrėta 2015-12-12]. Prieiga per internetą:
<http://www.iis.sinica.edu.tw/~swc/pub/game_traffic_analysis.html>

[DLN15] D. Lemire, L. Boytsov, SIMD Compression and the Intersection of Sorted Integers, Gegužė, 2015. Prieiga per internetą: <https://arxiv.org/pdf/1401.6399.pdf>

[DNP99] M. Degermark, B. Nordgren, D. Pink, IP Header Compression, RFC 2507, 1999.

[Jac90] Van Jacobson, Compressing TCP/IP Headers for Low-Speed Serial Links, Network Working Group, RFC1144, 2015.

[JND15] Jeff P., Nathan K., Daniel L., Vectorized VByte Decoding, International Symposium on Web AlGorithms, Liepa, 2015. Prieiga per internetą: <https://arxiv.org/pdf/1503.07387.pdf>

[Jon06] Jones T., Better networking with SCTP, Network Working Group, 2006. Prieiga per internetą: <http://www.ibm.com/developerworks/linux/library/l-sctp/>

[JK14] Asral Bahari Jambek, Nor Alina Khairi, Performance comparison of Huffman and Lempel-Ziv Welch data compression for wireless sensor node application, School of Microelectronic Engineering, Universiti Malaysia Perlis, 2014.

[Hed11] Mattias Hedén, SCTP An analysis of proposed implementations, Birželis, 2011.

[Huf07] D. A. Huffman, A method for the construction of minimum-redundancy codes, 2007.

[LTG05] J. Lau, Ed., M. Townsley, Ed., I. Goyret, Ed., Layer Two Tunneling Protocol - Version 3 (L2TPv3), 2005.

[OH03] Manuel Fradinho Duarte de Oliveira, M. Henderson, What online gamers really think of the Internet?, USA, 2007.

[PBP+09] Petlund, A. Beskow, P. Pedersen, J. Søgård Paaby, E. Griwodz, C. & Halvorsen, P., Improving SCTP retransmission delays for time-dependent thin streams, 2009. Prieiga per internetą: <http://folk.uio.no/paalh/publications/files/mtap09-SCTP.pdf>

[Pos81] J. Postel, Transmition Control Protocol, Information Sciences Institute University of Southern California, RFC 793, Rugsėjis, 1981. Prieiga per internetą: <https://tools.ietf.org/html/rfc793>

[Kle91] Anders Klemets, LZW Compression of interactive network traffic, Swedish Institute of Computer Science, 1991.

[Koo08] R. Koodli, Ed., Mobile IPv6 Fast Handovers, Liepa 2008.

[Nag84] J. Nagle, Congestion control in IP/TCP internetworks. RFC 896, Sausis, 1984.

[Npd13] The NPD Group: Report Shows Increased Number of Online Gamers and Hours Spent Gaming, Gegužė, 2013.

[PAC+11] V. Paxson, M. Allman, J. Chu, M. Sargent, Computing TCP's Retransmission Timer, Liepa, 2011. <https://tools.ietf.org/html/rfc6298>

[PHC+01] S. Pack, E. Hong, Y. Choi, I. Park, Jong-Sung Kim, D. Ko, Game Transport Protocol: A Reliable Lightweight Transport Protocol for Massively Multiplayer On-line Games (MMPOGs), Seoul, Korea, 2001.

[PWA11] H. Per, J. Wolfgang, A. Brunstrom, Recent Trends in TCP Packet-Level Characteristics, International Academy, Research, and Industry Association, 2011.

[SDM09] Mirko Suznjevic, Ognjen Dobrijevic, and Maja Matijasevic, MMORPG Player Actions: Network Performance, Session Patterns and Latency Requirements Analysis, Berlin, Croatia, 2009.

[SKR07] P. Svoboda, W. Karner, M. Rupp, Traffic Analysis and Modeling for World of Warcraft, USA, 2007.

[SS13] Jose Saldana, Mirko Suznjevic, Traffic of Online Games, Berlin, Germany, 2013.

[SSN+12] Jose Saldana, Luis Sequeira, Julian Fernandez Navajas, José Ruíz, Traffic optimization for TCP-based Massive Multiplayer Online Games, 2012.

[Ste07] R. Stewart, Ed., Stream Control Transmission Protocol, Rugsėjis, 2007.

[STP+11] R. Stewart, M. Tuexen, K. Poon, P. Lei, V. Yasevich, Sockets API Extensions for the Stream Control Transmission Protocol (SCTP), 2011. <https://tools.ietf.org/html/rfc6458>

[TKW05] B. Thompson, T. Koren, D. Wing., Tunneling Multiplexed Compressed RTP (TCRTP), RFC 4170, Lapkritis, 2005.

[VM09] SZABÓ G. VERES, S. MOLNÁR, On the impacts of human interactions in mmorpg traffic, 2009.

[VS14] Veluchamy Glory, Sandanam Domni, Compressing Inverted Index Using Optimal FastPFOR, Lapkritis, 2014.

[Wel84] Terry A. Welch, A Technique for High-Performance Data Compression, 1984.

[Wes13] M. West, RObust Header Compression (ROHC): A Profile for TCP/IP (ROHC-TCP), Internet Engineering Task Force (IETF), 2013.

[WKV+06] A. F. Wattimena, R. E. Kooij, J. M. van Vugt, O. K. Ahmed, Predicting the perceived quality of a first person shooter: the Quake IV G-model, 2016.

[ZA04] S. Zander, G. Armitage, Empirically Measuring the QoS Sensitivity of Interactive Online Game Players, Australian Telecommunications Networks, Sydney, Australia, Gruodis, 2004.