ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
«ҒҰМАРБЕК ДАУКЕЕВ АТЫНДАҒЫ АЛМАТЫ ЭНЕРГЕТИКА ЖӘНЕ БАЙЛАНЫС УНИВЕРСИТЕТІ»
Коммерциялық емес акционерлік қоғамы
Телекоммуникация және инновациялық технологиялар кафедрасы

«ҚОРҒАУҒА ЖІБЕРІЛДІ»
ТКИТ каф. меңгерушісі
PhD докторы, доцент
Қадылбекқызы Эльвира
(ғылыми дәрежесі, атағы, Т.А.Ж.)
_____ «____» _____2021ж.
(қолы)

ДИПЛОМДЫҚ ЖОБА

Тақырыбы: «Желілік трафикті жинау және терең зерттеу технологиялары»
Мамандығы: 5В071900 – «Радиотехника, электроника және телекоммуникациялар
Орындаған: Кисбек Санжар Ерболатулы Тобы: РЭТ(МТС)к-17-2
(Т.А.Ж.)
Ғылыми жетекшісі: доцент Жунусов Қанат Хафизович (ғылыми дәрежесі, атағы, Т.А.Ж.)
Кеңесшілер:
экономикалық бөлім бойынша:
доцент Салыкова Мадина Салыковна
(ғылыми дәрежесі, атағы, Т.А.Ж.)
_____ «____» _____2021ж.
(қолы)
өміртіршілігі қауіпсіздігі бойынша:
аға оқытушы Сапаев Тимур Михайлович
(ғылыми дәрежесі, атағы, Т.А.Ж.)
_____ «____» _____2021ж.
(қолы)
есептеу техникасын қолдану бойынша: профессор Чежимбаева К.С.
(ғылыми дәрежесі, атағы, Т.А.Ж.)
_____ «____» _____2021ж.
(қолы)
Нормобақылаушы: аға оқытушы, Накисбекова Балауса Рыскожаевна
(ғылыми дәрежесі, атағы, Т.А.Ж.)
_____ «____» _____2021ж.
(қолы)
Пікір беруші: _____
(ғылыми дәрежесі, атағы, Т.А.Ж.)
_____ «____» _____2021ж.
(қолы)

Алматы 2021

ҚАЗАҚСАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
«ҒҰМАРБЕК ДАУКЕЕВ АТЫНДАҒЫ АЛМАТЫ ЭНЕРГЕТИКА ЖӘНЕ БАЙЛАНЫС УНИВЕРСИТЕТІ»
Коммерциялықемесакционерлікқоғамы

Телекоммуникация және ғарыштық инженерия институты
Телекоммуникация және инновациялық технологиялар кафедрасы
5В071900–Радиотехника, электроника және телекоммуникациялар мамандығы

Дипломдықжобаныорындауғаберілген

ТАПСЫРМА

Студент: Кисбек Санжар Ерболатулы_____

Жобаның тақырыбы«Желілік трафикті жинау және терең зерттеу технологиялары»

2020 ж. «27» қазан № 217 университет бұйрығымен бекітілді.

Аяқталған жобаны тапсыру мерзімі «25» мамыр 2021 ж.

Жобаға алғашқы деректер (талап етілетін зерттеу (жоба) нәтижелерінің параметрлері және зерттеу нысанының алғашқы деректері):

Деректер желілерін жаңарту тез және экономикалық тұрғыдан қымбатқа түсетіні анық, сондықтан байланыс операторлары DTM шешімдерін (Data Traffic Management) қолданады: PCRF (Policy and Charging Rules Function) және DPI (Deep Packet Inspection).

_____

Диплом жобасындағы әзірленуі тиіс мәселелер тізімі немесе диплом жобасының қысқаша мазмұны:

Кіріспе_____

1. Желілік трафикті терең талдау жүйелері

2. Жоғары жылдамдықты фреймворктарды басып алу және өңдеу дестетері__

3. Дестелерді басып алу өнімділігі мен трафикті жіктеуді тестілеу_____

4. Өмір тіршілік қауіпсіздігі _____

5. Жобаның техникалық-экономикалық негіздемесі _____

Қорытынды_____

Әдебиеттер тізімі_____

_____

_____

_____

Графикалық материалдардың (міндетті түрде дайындалатын сызбаларды көрсету) тізімі:

1. «Invest» шағын компаниясының желісіндегі IOT құрылғылары мен компьютерлерін қосу сұлбасы

2. Компьютерлер мен IOT құрылғыларының интернет-трафигінің кестесі

Негізгі ұсынылатын әдебиеттер:

1 Carrier Big Data, Deep Packet Inspection, Analytics, and Data as a Service 2015 - 2020 // [Research and Markets] URL: https://tinyurl.com/ybudt6km (дата обращения 31.12.2017).

2 Опрос «Мобильная связь в Казахстане» // [Официальная страница компании 4Service Group] URL: https://tinyurl.com/y7xwdyht (дата обращения 31.12.2017).

3 Kevin Roebuck. Deep Packet Inspection (DPI): High-Impact Strategies - What You Need to Know: Definitions, Adoptions, Impact, Benefits, Maturity, Vendors // Australia: Emereo Publishing, 2012, ISBN 1283761904 (ISBN13: 978-1-28376-190-1).

4 Oliver M. Heckmann. The Competitive Internet Service Provider: Network Architecture, Interconnection, Traffic Engineering and Network Design. USA, New York: John Wiley & Sons, 2007, ISBN: 978-0-470-03004-2 – 398 pp

Жоба бойынша жобаның бөлімдеріне қатысты белгіленген кеңесшілер

Бөлімдері	Кеңесшілері	Мерзімі	Қолы
Экономика	Салыкова М.С.	13.04. 2021ж.	
Ө.Т.Қ.Н.	Сапаев Т.М.	13.04. 2021ж.	
Негізгі бөлім	Жунусов К.Х.	14.03. 2021ж.	
Есептеу техникасы	Тұманбаева К.Х.	16.04. 2021ж.	
Нормабақылаушы	Накисбекова Б.Р.	.	

Аңдатпа

Бұл дипломдық жобада жоғары пакеттік жылдамдықта желілік пакеттерді түсіру және желілік трафикті жіктеу технологияларын қарастырады. Тиісті ашық бастапқы шешімдер сыналды. Өмір тіршілік қауіпсіздігі тұрғысынан қажетті мәселелер талданды. Экономикалық бөлімде бағдарламалық өнімді әзірлеу бағасы есептелді.

Аннотация

В данном дипломном проекте рассматриваются технологии захвата сетевых пакетов на высоких пакетных скоростях и классификации сетевого трафика. Протестированы соответствующие решения с открытым исходным кодом. В части безопасности жизнедеятельности проанализированы необходимые вопросы. В экономической части рассчитывалась цена разработки программного продукта.

Abstract

This thesis project examines technologies for capturing network packets at high packet rates and classifying network traffic. Corresponding open source solutions have been tested. In terms of life safety, the necessary issues have been analyzed. In the economic part, the price of software product development was calculated.

Мазмұны

Кіріспе

Қазіргі уақытта көптеген шетелдік терең дестетік инспекция (DPI) жүйелері бар. Әдетте, олардың барлығы коммерциялық, айтарлықтай ақшалай құны бар және, тиісінше, жабық. Сарапшылардың болжамдарына сүйенсек, келесі бірнеше жылда DPI жүйелерінің экономикалық нарығы қарқынды өсуді бастан кешуде. Сондай-ақ, бұл жүйелер қарапайым нәрсе емес, күрделі құрылымнан және өзара әрекеттесетін көптеген компоненттерден тұратыны анық. Дипломдық жобаның мақсаты ашық бастапқы кодтарға негізделген кейбір компоненттерді талдау болып табылады, бұл жүйелер туралы мәліметтер: желілік дестетерді жоғары дестетік жылдамдықта басып алу және протоколдар мен желілік трафик ағындарын жіктеу.

Осы мақсатқа жету үшін желілік трафикті терең талдау жүйелерінің мақсатын, жұмыс принципі мен қолдану сценарийлерін түсіну, Осы саладағы қолданыстағы шешімдерді қарастыру, мүмкін болса, олардың архитектурасын, ерекшеліктерін талдау, сонымен қатар нақты тестілеу жағдайларына жақын болу керек.

Желілік дестеті алу және өңдеу кезінде аппараттық деңгейде және операциялық жүйеде болып жатқан процестерді, қойылған шектеулерді және оларды айналып өту әдістерін түсіну желілік технологиялар мен операциялық жүйелердің желілік дестесінің даму бағыты туралы, сондай-ақ желілік трафикті жоғары дестетік жылдамдықта өңдеу кезінде туындайтын мәселелер туралы түсінік береді, бұл сөзсіз пайдалы болады.осы бағытта әрі қарай зерттеу.

1 Желілік трафикті терең талдау жүйелері

1.1 Желілік трафикті талдау және басқару жүйелері нарығын дамыту перспективалары

Адамдар өміріндегі байланыстың маңыздылығы артып келе жатқандықтан, интернетті және ұялы байланысты пайдаланушылар саны артып, сәйкесінше көрсетілетін қызметтердің сапа деңгейіне қойылатын талаптар артып келеді. Интернет-провайдерлер алдында, әсіресе екінші деңгейлі, жеке байланыс желілері жоқ және/немесе бастапқы Трафиктен желілік трафикті сатып алу алдында үлкен және өсіп келе жатқан интернет-трафикті талдау және басқару міндеті пайда болды. Деректер желілерін жаңарту тез және экономикалық тұрғыдан қымбатқа түсетіні анық, сондықтан байланыс операторлары DTM шешімдерін (Data Traffic Management) қолданады: PCRF (Policy and Charging Rules Function) және DPI (Deep Packet Inspection).

PCRF – бұл EPC (Evolved Packet Core) құрамына кіретін, желілік ядро деңгейінде жұмыс істейтін және әр пайдаланушы туралы ақпараты бар бағдарламалық жасақтама (БЖ). PCRF OSS (Operations Support Systems) эксплуатациялық қолдау жүйесінен, трафикке қызмет көрсету саясатын автоматты басқаруды, әрбір жеке белсенді пайдаланушы үшін QoS (Quality of Service) баптауды қолдай отырып, желінің ағымдағы жағдайы туралы ақпаратты агрегаттайды. PCRF сонымен қатар тарифтеу, ақы төлеу, пайдаланушы статистикасын жүргізу қызметтерімен біріктірілуі мүмкін.

DPI - бұл OSI (Open Systems Interconnection) моделінің 7 деңгейіне дейін желідегі барлық трафикті нақты уақыт режимінде талдайтын, статистикалық мәліметтерді жинақтайтын, желілік дестетерді олардың мазмұны негізінде тексеретін және сүзетін технология. DPI сонымен қатар желілік дестетердің мазмұнын "жылдам" өзгертуге мүмкіндік береді. DPI қолдана отырып, ISP (Internet Service Provider) желілік трафикті протоколдар мен қосымшалар бойынша жіктеуге, әр түрлі трафиктің басымдылығын басқара отырып және кез-келген уақытта белгілі бір өткізу қабілетіне кепілдік бере отырып, өткізу қабілетін тиімді жоюға мүмкіндік алады. Мысалы, HTTP, HTTPS, RTP, RTSP, IGMP, SIP, H.323, Skype сияқты протоколдарға, ал ең кішісі – BitTorrent, µTP.

Дипломдық жобаның атауынан белгілі болғандай, содан кейін тек екінші нұсқа толығырақ қарастырылады.

Сарапшылардың болжамы бойынша, DPI жаһандық нарығы 2020 жылға қарай CAGR (Compound Annual Growth Rate) 2014 жылдан бастап 2020 жылға дейін 30,2% - ға ұлғаюымен 4,71 млрд.USD-ға бағаланатын болады. Негізінен, DPI-ді интернет провайдерлері, ұялы байланыс операторлары пайдаланады.

DPI нарығының өсуі, ең алдымен, пайдаланылатын смартфондар санының өсуіне және кең жолақты деректер сұранысына байланысты, бұл дәстүрлі дауыстық байланыс туралы айту мүмкін емес, бұл TSOP (Ортақ

пайдалану телефон желісі) немесе ұялы телефон болсын, жыл сайын кірісі төмендейді. ҚР-да мобильді интернетті пайдаланушылар саны айтарлықтай өсті. Сонымен, егер 2016 жылы смартфон иелерінің 80% мобильді интернетті пайдаланса, 2017 жылы - 83,3%.

Сол сияқты, пайдаланушылар тұтынатын және жасайтын трафик көлемі азаймайды, бірақ үлкен қарқынмен өсуде. Мысалы, 2017 жылы айына 6 ГБ – тан асатын мобильді интернетті пайдаланушылардың үлесі 33% - ды құрады, 2016 жылы бұл көрсеткіш-27%. Ай сайын 1 ГБ-тан аз жүктейтін пайдаланушылар үшін ол 12% - ға азайды.

Осылайша, смартфон иелері мобильді интернетті белсенді қолдана бастайды, бұл ұялы байланыс операторларына жүктемені арттырады.

DPI интернет-провайдер немесе байланыс операторы үшін типтік мәселені шешуге мүмкіндік береді, мысалы, 20 Мб/с үшін 2500 пайдаланушы, бұл жалпы 50 (2500*20) Гб/с арнада бар-жоғы 40 Гб/с, өткізу қабілеттілігін әртүрлі сүзгілер, кэштер және басымдықтар арқылы оңтайландырады. Жаңа абоненттердің Болашақ ағынымен күресу үшін, Quality of Experience (QoE) жоғары деңгейде сақтай отырып, желі инфрақұрылымы дамып, қуаттылық өскен сайын операторлар DPI технологиясына жиі жүгінеді.

Бұл факторлар алдағы бірнеше жылда DPI шешімдеріне өсіп келе жатқан сұранысты анықтайды.

DPI шешімдерін ұсынатын қазақстандық компаниялардың талдауы олардың аз санын, пайдаланылатын технологиялар, Статистика, шамамен баға диапазоны туралы қандай да бір егжей-тегжейдің жоқтығын көрсетті.

1.2 Әр түрлі DPI қолдану сценарийлері

Dpi жүйесі, аббревиатураның шифрынан көрініп тұрғандай, ол арқылы өтетін барлық желілік дестетерге терең талдау жасайды. "Терең" ұғымы OSI желілік моделінің жоғарғы деңгейлерінде (layer 2, data link) және ең төменгі деңгейге дейін (layer 7, application) дестетерді талдауды білдіреді. Желілік трафикті белгілі бір стандартты шаблондар бойынша талдаудан басқа, дестетің белгілі бір қолданбаға тиесілігін нақты немесе нақты дәлдікпен анықтауға болады, мысалы, тақырыптардың форматы мен мазмұны, порт нөмірлері және т.б. арқылы, dpi жүйесі трафиктің мінез-құлық (эвристикалық) талдауын жүзеге асырады, бұл сізге белгілі тақырыптар мен деректер құрылымын пайдаланбайтын қосымшаларды тануға мүмкіндік береді. Бұған айқын мысал – P2P принципі бойынша жұмыс істейтін және/немесе BitTorrent протоколын қолданатын қосымшалар. Осындай қосымшалардың трафигін сәйкестендіру үшін Source_IP:Port – Destination_IP:Port, дестетің мөлшері, уақыт бірлігіне жаңа сессияларды ашу жиілігі және т.б. сияқты белгілері бірдей желілік дестетердің белгілі бір реттілігін талдау жүзеге асырылады.

Тәжірибеде іске асырылатын негізгі DPI қолдану сценарийлері:

- трафикті талдау және жіктеу, мониторинг және трендтер;
- провайдерді оңтайландыру;

- ресурстарға қол жеткізуді шектеу (ақ және қара тізімдер);
- пайдаланушылардың мінез-құлқын талдау;
- желілік трафиктің басымдылығы;
- пайдаланушылар арасында арнаны икемді бөлу;
- - кэштеу;
- пайдаланушыларға кодты, мысалы, HTML бетіне қайта бағыттау немесе ендіру арқылы хабарлау
- қорғау, трафикті ұстап қалу, ЖІШЖ алдын алу сүзгі.

1.2.1 Трафикті талдау және жіктеу, мониторинг және трендтер

DPI міндеттеріне қол қоюлар мен белгілердің кез-келген жиынтығы негізінде желілік трафикті талдау және жіктеу, белгілі бір хаттаманың, қосымшаның немесе тұтастай пайдаланушының мінез-құлық модельдерін құру, нақты уақыт режимінде әртүрлі есептер құру жатады.

Skype, WhatsApp, Viber, BitTorrent, SIP, YouTube және т.б. сияқты қосымшалар мен қызметтерді пайдаланушының жалпы трафик ағынынан бөлектеу тарификацияны икемді түрде саралауға және оны биллинг жүйелерінде пайдалануға мүмкіндік береді.

Оператордың желісіндегі желілік трафикті осындай талдау желідегі жүктемелерді болжау және желінің кеңеюін одан әрі жоспарлау немесе өткізу қабілетін арттыру мақсатында ұзақ уақыт бойы егжей-тегжейлі статистиканы жинауға мүмкіндік береді.

1.2.2 Трафиктің басымдылығы

DPI жүйесін желідегі белгілі бір трафиктің басымдылығын бақылау үшін қолдануға болады. Мысалы, кейбір немесе барлық пайдаланушыларға, хаттамаларға немесе қосымшаларға оператордың желісінде тұрақты жылдамдықты және/немесе төмен кідірісті қамтамасыз ету.

Интернет-провайдерлердің бас ауруларының бірі-P2P трафигі, ол барлық өткізу қабілеттілігін "ұстап" алады, осылайша провайдердің арнасын жүктейді, бұл басқа протоколдарды немесе қосымшаларды (VoIP, video, HTTP және т.б.) пайдалану кезінде байланыс сапасына әсер етеді.

Бұл жағдайда P2P трафигін анықтап, оны шектеу керек, бұл DPI мүмкін етеді.

DPI-де әртүрлі протоколдардың тақырып мәндерін өзгерту мүмкіндігі бар: IP дестеінің ToS тақырыбы (DSCP және ECN мәндері); VLAN-(IEEE 802.1 p) және Q-in-Q-дестетеріндегі PCP тақырыбы (IEEE 802.1 ad); MPLS дестеінің traffic Class тақырыбы (QoS және ECN мәндері). QoS деңгейін ұстап тұру үшін аталған тақырыптарды маршрутизаторлар мен traffic shaper оқиды.

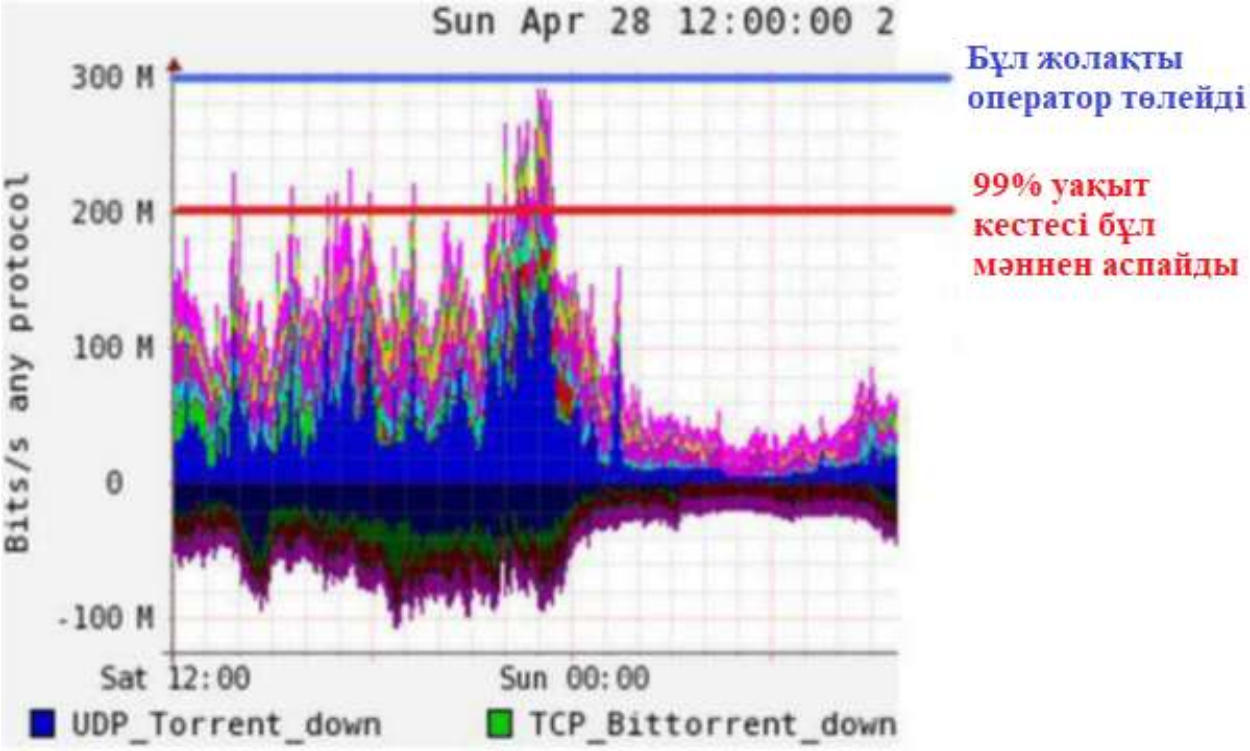
1.2.3 Uplink провайдерді оңтайландыру

Жай сөзбен айтқанда, Uplink оператор Интернетке қол жеткізетін арна деп аталады. Онсыз барлық пайдаланушылар оператордың желісіне қосылатын еді және тек.

Uplink – бұл ай сайынғы оператордың шығындарының көп бөлігін құрайтын оператор үшін ресурстардың бір түрі. Uplink өткізу қабілеті неғұрлым кең болса, оператор өз абоненттеріне жоғары жылдамдықты ұсынуға дайын. Uplink модернизациясы өте қымбат және кейде мүмкін емес операция болып табылады, сәйкесінше, бұл жағдайда qos жоғары деңгейіне сәйкес болу үшін оператордың желісіндегі трафикті оңтайландыру қажет.

Мысал ретінде кездейсоқ интернет-провайдердің статистикасын алайық. 1.1-суреттен көріп отырғанымыздай, провайдер 300 Мб/с өткізу қабілеттілігін төлейді, ал уақыттың көп бөлігі трафик 200 Мб/с аспайды. сондай-ақ, UDP_Torrent_down трафигі үлкен жүктеме тудырады, оны QoS деңгейін жоғалтпай шектеуге болады.

Арнаның өткізу қабілетін айқын оңтайландыру бар, оны DPI жүйесін қолдана отырып жасауға болады.



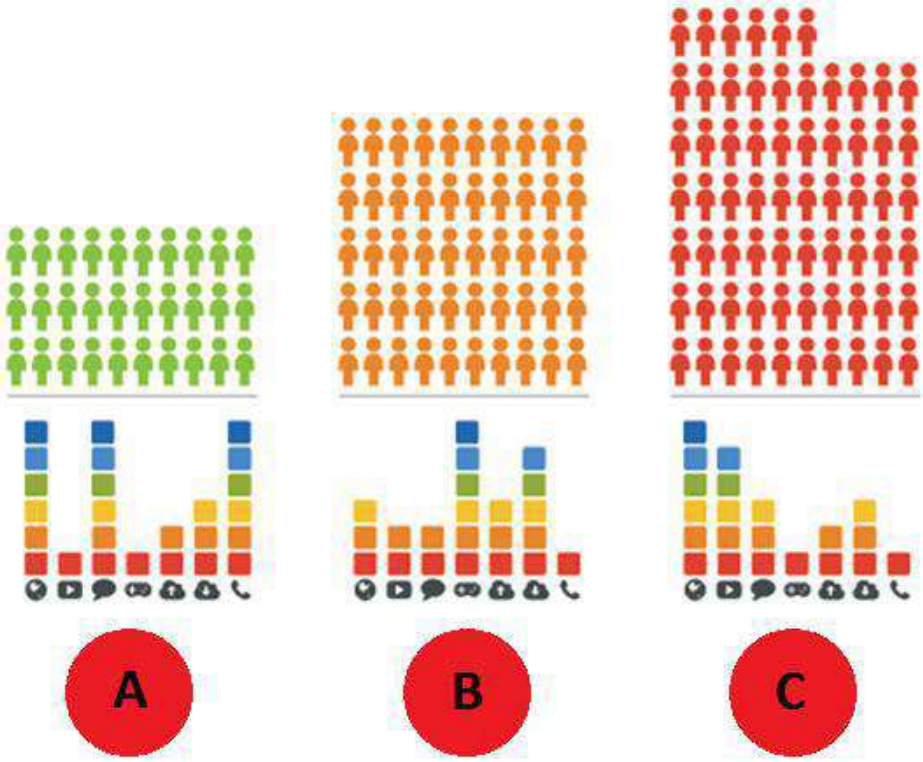
1.1 сурет – Тәулік ішінде Torrent, BitTorrent және т. б. қосымшалармен арнаның өткізу қабілеттілігін пайдалану статистикасы

1.2.4 Пайдаланушылар арасында арнаны бөлу

DPI жүйесін қолдана отырып, трафикті тек протоколдар және/немесе қосымшалар бойынша ғана емес, сонымен қатар пайдаланушылар бойынша да жіктеуге болады, бұл тиісті пайдаланушылар топтары үшін әртүрлі деректер жоспарларын құруға мүмкіндік береді.

Мысалы, пайдаланушылардың топтарға жіктелуін қарастырыңыз (1.2-суретті қараңыз). "А" тобына тұрақты сапалы дауыстық және мәтіндік байланыс, сондай-ақ Интернетте тегіс серфинг қажет болатын корпоративті

клиенттер кіреді делік. "B" тобына кез-келген уақытта ойын серверлеріне төмен кідірісті және жылдам жүктеу мүмкіндігін қажет ететін, мысалы, сатып алынған ойынды немесе оған жаңартуды қажет ететін әртүрлі мультипликативті желілік ойындардың ойыншылары кіреді. "C" тобына интернеттегі көп уақытты веб-серфинг, бейне көру және байланыс үшін өткізетін провайдер пайдаланушыларының көп бөлігі кіреді. Тиісінше, трафиктің әр түрі үшін әр топтың өз басымдықтары бар, олар топ ішінде де, топтар арасында да әрекет ете алады.



1.2 сурет – Пайдаланушыларды трафиктің әртүрлі басымдықтары бар топтар бойынша жіктеу

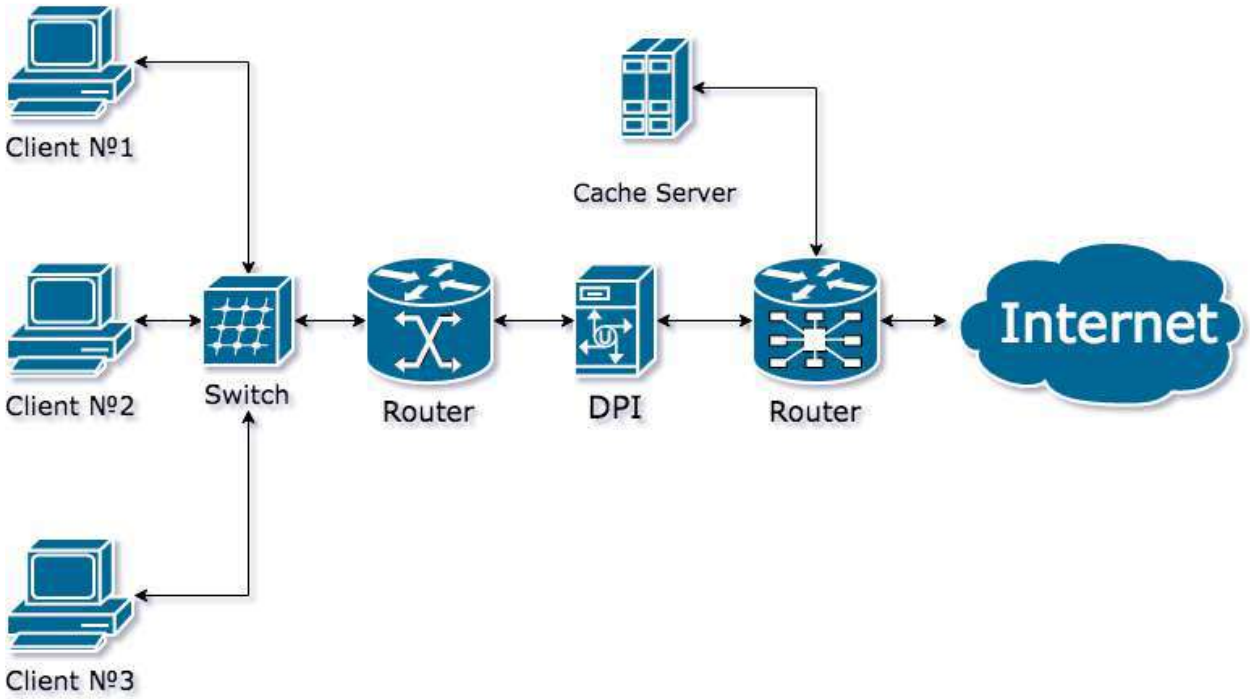
Бұл трафик басымдықтарын провайдер желісіндегі жүктемені икемді теңдестіру, соңғы пайдаланушылар үшін әртүрлі тарифтер құру үшін пайдалануға болады. Сондай-ақ, жіктеу деректерін әртүрлі аналитика үшін пайдалануға болады.

1.2.5 Кэштеу

Интернет - бұл ақпараттың үлкен көзі, өкінішке орай, аз адамдар тиімді пайдаланады. Интернетте, адамдардың өміріндегідей, танымал және өте жиі және сирек қолданылатын деректер бар. Әлбетте, ақпарат неғұрлым танымал болса, соғұрлым жиі хабарласады. Бұл корреляцияны провайдердің арнайы кэш-серверлерінде қолдануға болады, олар ұқсас ақпаратты анықтайды және оны өздері жазады.

Тиісінше, пайдаланушы осы ақпаратқа провайдер желісінен тыс кез-келген ресурста келесі рет жүгінген кезде, оны қайтадан жүктеместен және uplink-ті жүктеместен оны жергілікті сервер кэшінен (cache server) пайдаланушыға беруге болады (1.3-суретті қараңыз).

Мысал ретінде CDN (Content Delivery Network) және оның негізінде GGC (Google Global Cache) келтіруге болады. GGC танымал YouTube видео хостинг қызметінің ресурстарын кэштеуге арналған арнайы бағдарламалық жасақтамасы бар бірнеше сервер.



1.3 сурет – Cache Server-мен бірге DPI қолдану мысалы

1.2.6 Пайдаланушылар туралы статистикалық деректерді жинау

Әр адам өзінше ерекше немесе, кем дегенде, саналы түрде осындай болуға тырысады және бұл оның жаһандық желідегі әрекеттерінен көрінеді. Пайдаланушы жыл бойы, күн сайын тәуліктің белгілі бір уақытында жеке құрылғының (планшет, ноутбук, смартфон және т.б.) көмегімен өзінің сүйікті браузерін пайдалана отырып, әртүрлі интернет ресурстарға (жаңалықтар сайттарына, әлеуметтік желілерге және т. б.) барады, бір немесе бірнеше мессенджерлерде сөйлеседі, кейбір бағдарламаларда жұмыс істейді және/немесе көп ойыншы ойындарын ойнайды.

DPI жүйесі одан әрі талдау үшін ұқсас иесіздендірілген статистикалық деректерді жинауға мүмкіндік береді (1.4-суретті қараңыз).

Top Visited Domains

« ‹ 1 2 3 4 5 6 7 › »

DOMAIN	UNIQUES	VISITS	% USERS	% VISITS	VISITS/UNIQUE
yandex.ru	34,735	1,085,863	49.42	9.64	31.26
vk.com	29,948	4,964,305	42.61	44.05	166
google.com	19,752	490,821	28.1	4.36	24.85
mail.ru	13,319	93,778	18.95	0.832	7.04
google.ru	12,290	189,028	17.49	1.68	15.38
youtube.com	6,781	127,589	9.65	1.13	18.82
ok.ru	6,767	41,339	9.63	0.367	6.11
ebanoe.it	5,968	63,472	8.49	0.563	10.64
lurkmore.to	5,494	17,898	7.82	0.159	3.26
pikabu.ru	4,780	52,200	6.8	0.463	10.92
TOTAL	70,287	11,269,469	100	100	160

1.4 сурет – Интернет-провайдер пайдаланушыларының белгілі бір интернет ресурстарға кіру статистикасы

1.2.7 Абоненттер хабарламасы

Интернет заманауи адамның өміріне тығыз кіргені ешкімге құпия емес. Күні бойы адамдар көбінесе отбасымен және достарымен сөйлесу үшін интернетті қолдана отырып, назарын аударады; жаңалықтар мен мақалаларды оқу үшін; мысықтармен түрлі суреттер мен бейнелерді қарау үшін және т. б.

DPI жүйесін қолдана отырып, оператор хабарламаларды HTML парақтарына ендіруге, пайдаланушыны ақпараттық сайттарға бағыттауға мүмкіндік алады. Хабарламалардағы ақпарат оператордан хабарландыруларды (осы пайдаланушының тарифтік жоспарының өзгеруі, желідегі техникалық жұмыстар, арнайы ұсыныстар және т.б.), сондай-ақ әртүрлі мемлекеттік құрылымдардың төтенше хабарламаларын қамтуы мүмкін.

1.2.8 Ресурстарға қол жеткізуді басқару (ақ және қара тізімдер)

ҚР аумағында немесе қандай да бір басқа елде өз қызметтерін ұсынатын кез келген байланыс операторы, ол қаншалықты объективті емес және шындыққа сәйкес келмейтін болса да, тиісті елдің заңнамасын сақтауы тиіс екені даусыз. Қазіргі әлемдегі ақпарат құнды ресурс болып табылады және бір IP мекенжайы мен бір веб – ресурс арасындағы сәйкестік ұзақ уақыт бойы дұрыс емес, сондықтан URL мекен-жайына байланысты сүзгілеуді пайдалану оператор үшін ең дұрыс шешім болып табылады.

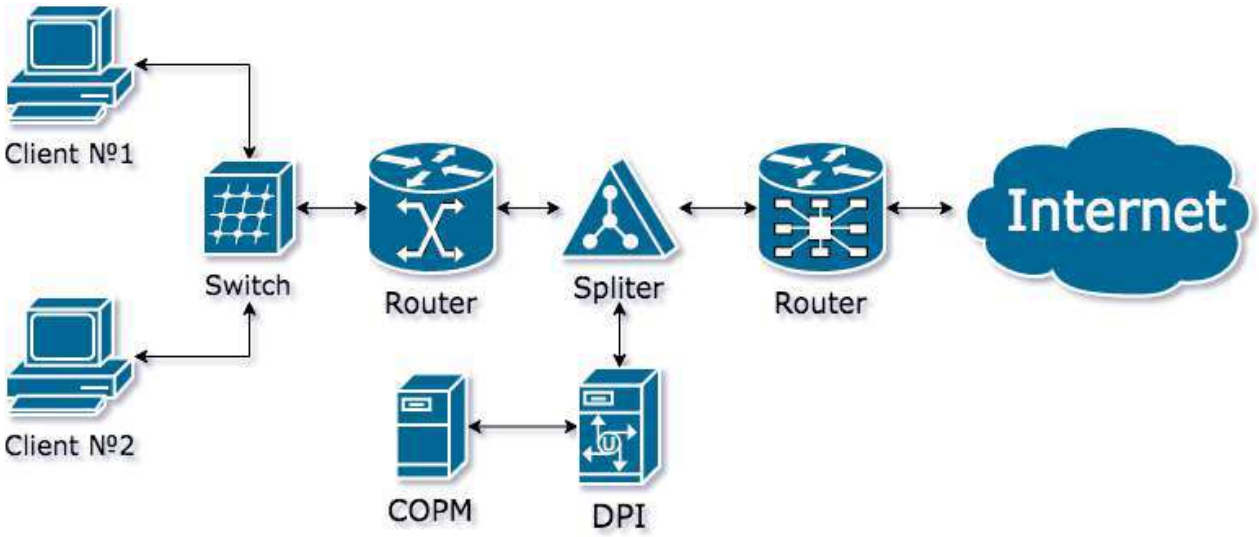
DPI жүйесі HTTP тақырыптарын талдай алады және жиналған мәліметтер негізінде тиісті ресурстарға қол жеткізуді шектеу үшін келесі әрекеттерді орындай алады. HTTPS жағдайында сертификаттарды ұстап қалу және меншікті жасау үшін MITM деп аталатын нұсқа бар, бірақ бұл әрекеттер соңғы пайдаланушы үшін қауіпті болып саналады, шифрлау принциптеріне сәйкес келмейді және провайдерлер өте сирек қолданылады.

1.2.9 Желіні қорғау және ЖІШШЖ-не қосымша

DPI жүйесінің көмегімен желі күйіндегі және желілік трафиктегі әртүрлі ауытқуларды анықтауға болады, өйткені ол арқылы өтетін барлық желілік трафикті талдайды. Мысалы, SMTP трафигінің өте көп мөлшері бір желі түйінінен пайда болған жағдайда болжамды спам-ботты бұғаттау.

Сондай-ақ, DPI жүйесі TCP SYN Flood (TCP хаттамасы бойынша қосылу сұраныстары), UDP Flood, UDP Flood DNS сияқты желілік шабуылдардан қорғауға мүмкіндік береді.

DPI-дің тағы бір қосымша функциясы жедел-іздестіру іс-шараларының (ЖІШШЖ) функцияларын қамтамасыз етуге арналған техникалық құралдар жүйесіне өзіндік "көмек" болуы мүмкін (1.5-суретті қараңыз). Белгілі бір қосымшалар мен қызметтердің (Twitch, YouTube, Torrent және т.б.) трафигін болдырмау үшін DPI қолдана отырып, ЖІШШЖ серверлерінде берілетін және сақталатын трафик көлемін едәуір азайтуға болады.



1.5 сурет – ЖІШШЖ-мен бірге DPI қолдану сұлбасы

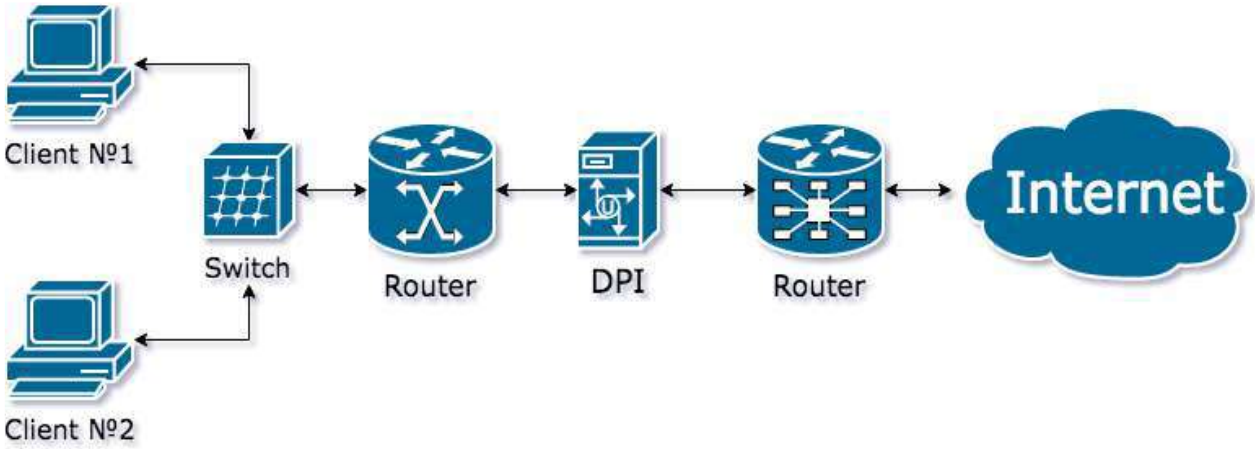
1.3 DPI қосылу сұлбалары

DPI жүйелері белсенді және пассивті болып бөлінеді. Белсенді DPI желінің белгілі бір аймағында "үзілісте" орнатылады және барлық трафикті өздері арқылы өткізеді, сондықтан, әдетте, мұндай DPI жұмысына көп көңіл бөлінеді, өйткені жүйе істен шыққан жағдайда желі қосылымы үзіледі. Белсенді DPI желілік дестетерге тікелей әсер ете алады.

Пассивті DPI айналы трафикті талдайды және желінің ақаулыққа төзімділігіне әсер етпейді. DPI-ді қолданудың бұл жағдайы ең Бюджеттік болып табылады. Мұндай DPI жұмыс сұлбасы хаттамаларды іске асырудың кейбір ерекшеліктерін қолдана отырып, нәтижеге негізделген: белгілі бір шығыс сұранысты немесе трафикті анықтаған кезде, DPI жауапты бастапқы серверден жауап алғанға дейін географиялық жақындығына байланысты бастапқы түйінге жібереді.

1.3.1 "Үзілісте" орнату сызбасы

DPI жүйесінің барлық мүмкіндіктерін іске асыру үшін шекаралық маршрутизатордан кейін оператордың сыртқы арнасының (uplink) үзілісінде орнату сұлбасын пайдалану қажет (1.6-суретті қараңыз).

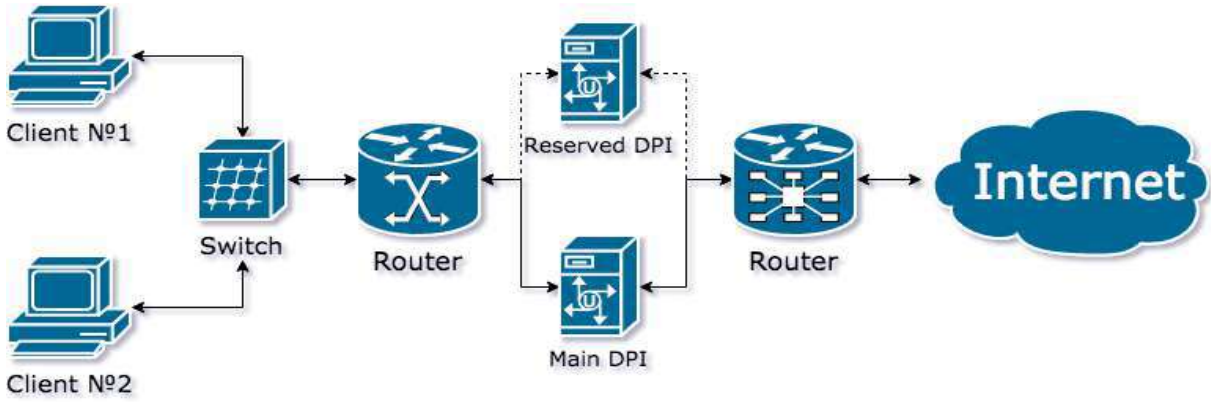


1.6 сурет – DPI «үзілісте» қосылу сұлбасы

Жоғарыда айтылғандай, мұндай сұлбаны қолданған кезде барлық желілік трафик dpi арқылы өтеді, бұл басымдық, пішіндеу, кэштеу және т.б. сияқты функцияларды жүзеге асыруға мүмкіндік береді. Бұл кемшілікті екі жолмен шешуге болады:

- DPI Bypass құрылғысын пайдаланыңыз, егер DPI-де апат болса немесе онымен байланыс үзілген болса, барлық желілік трафикті өзі арқылы өткізе бастайды;

- DPI резервін жасаңыз (1.7-суретті қараңыз), бұл DPI жүйесінің құнын екі есе арттырады.

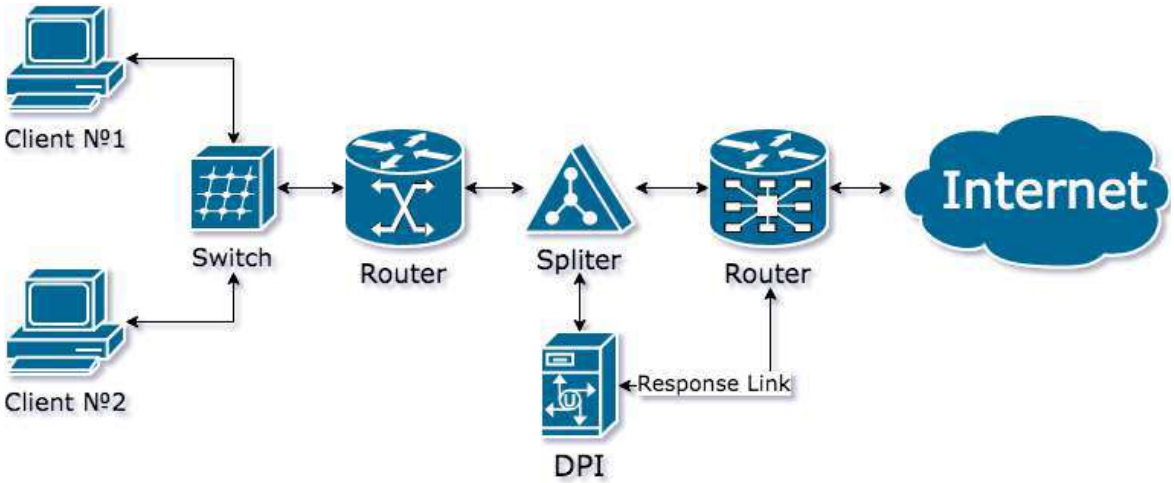


1.7 сурет – Резервтеуді қолдана отырып, DPI-ді "үзіліске" қосу сұлбасы

1.3.2 Трафикті көрсету сұлбасы

Желілік трафикті көрсету - бұл барлық трафикті жеке интерфейске бағыттау үшін қайталау. Трафикті көрсету коммутатордың немесе маршрутизатордың немесе оптикалық сплиттердің көмегімен SPAN порттары арқылы жүзеге асырылады.

Пассивті сұлбада (1.8-суретті қараңыз) DPI қосылымы bypass және резервтік шешімдерді қолданудың қажеті жоқ, өйткені DPI істен шығуы желіге әсер етпейді. Белсенді сұлбамен салыстырғанда, бұл опция желілік трафикті толық көлемде талдауға, кэштеу серверлерін қосуға, ЖІШШЖ жабдығына түсетін деректерді оңтайландыруға мүмкіндік береді.
Мүмкіндіктерді шектейтін пайдаланушылар ретінде пайдаланушылардың белгілі бір ресурстардың сұраныстарына ертерек жауап қалады, оны басқа ресурстарға бағыттау үшін пайдалануға болады және DPI-ден келетін желілік дестетерді талдау және оларды бұғаттау арқылы сұрау салушы Тарапқа оңай түседі, өйткені жауап қажетті сұралған ресурстардан пайдаланушыға келеді, бірақ үлкен кідірістерге байланысты операциялық жүйе оларды реле ретінде қабылдайды және тастайды.

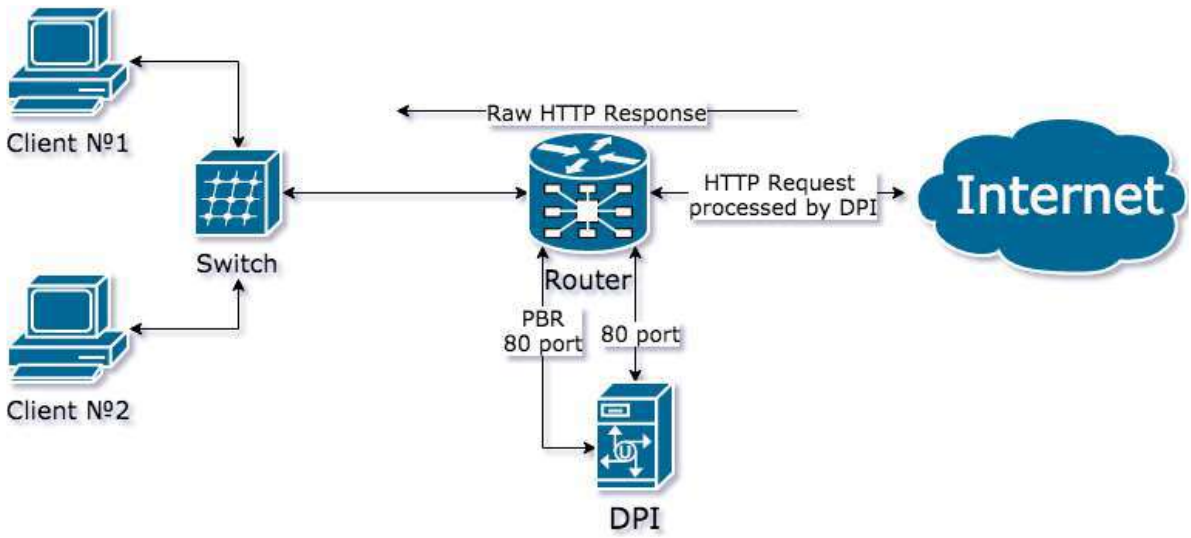


1.8 сурет – Пассивті DPI қосылу сұлбасы

1.3.3 Қосымша қосылу сұлбалары

Мысалы, тек 80 порттағы (HTTP) желілік трафикті талдау және оған тікелей әсер ету қажет болған жағдайда policy based routing (PBR) негізіндегі асимметриялық сұлбаны қолдануға болады. 1.9-суретте осы режимдегі DPI жұмысының визуалды сұлбасы көрсетілген. Пайдаланушылардан 80-ге шығатын трафик порт HTTP сұрауын өңдейтін, оны интернет-ресурсқа бағыттайтын немесе провайдер желісіндегі пайдаланушыға жауап қалыптастыратын DPI серверіне бағытталады.

Мұндай қосылу кезінде өнімді жабдық қажет емес, өйткені трафиктің бұл түрі өткізу қабілеттілігінің аз бөлігін алады. Бұл сұлбаны пайдалану кезінде бүкіл желілік трафик бойынша көлемді статистиканы алу мүмкін емес екені түсінікті.



1.9 сурет – Policy based routing негізіндегі пассивті DPI қосылу сұлбасы

2 Жоғары жылдамдықты фреймворктарды басып алу және өңдеу дестетері

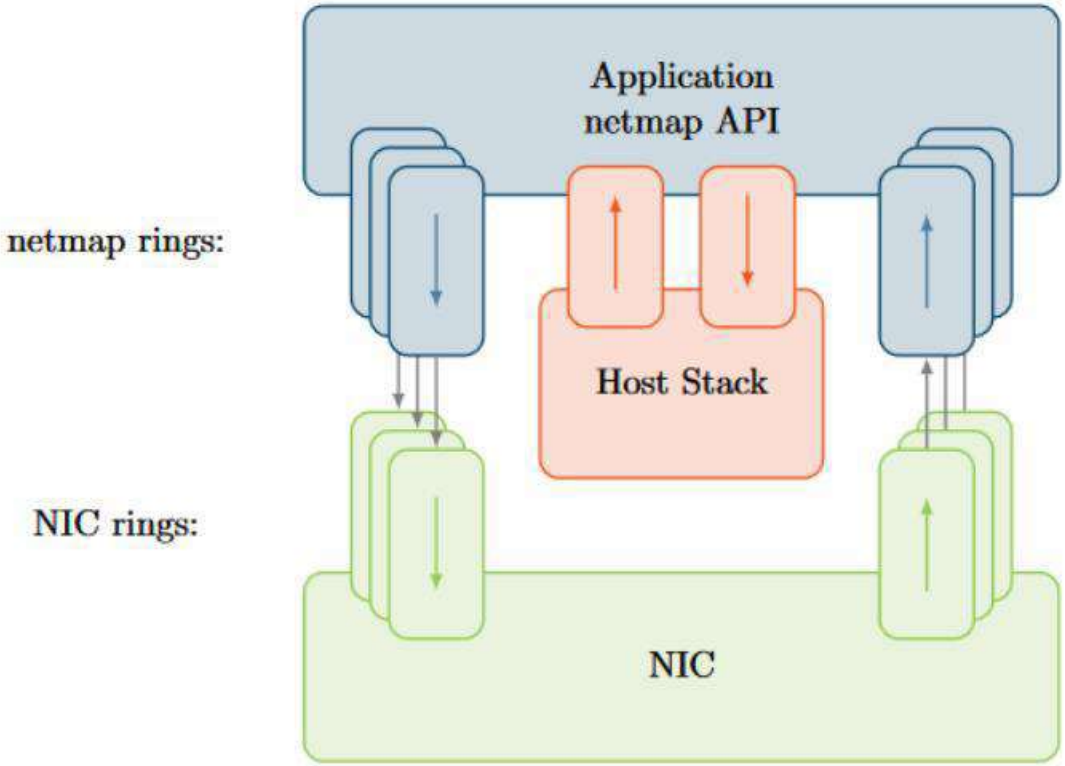
NETMAP, PF_RING ZC, DPDK сияқты жоғары жылдамдықты желілік дестетерді басып алуға арналған танымал фреймворктарды қарастырамыз.

2.1 NETMAP

2.1.1 Шолу

2.1-суретте желі арқылы деректерді беру үшін NETMAP API қолданатын қосымшаның өзара әрекеттесуінің жеңілдетілген сұлбасы көрсетілген. Бағдарлама мен желілік карта арасындағы мәліметтер алмасу NETMAP қоңырау буферлері (rings buffers) және желілік карта деп аталатын деректер құрылымдары арқылы жүзеге асырылады.

Кіріс және шығыс трафик жеке сақиналарды пайдаланады: бір немесе бірнеше сақиналы буферлер әр бағыт үшін қол жетімді болуы мүмкін. NET MAP қолданатын қосымшаны орындау кезінде хосттың желілік стегі байланыстан ажыратылады және желілік адаптер NETMAP режиміне өтеді.



2.1 сурет – NETMAP API қолданған кезде қосымшаның, ОЖ мен желілік картаның өзара әрекеттесуінің жеңілдетілген сұлбасы

Дестетерді қосымшадан хост стекіне және керісінше екі арнайы хост буфері арқылы беруге болады (host rings buffers). Хосттың сақиналы буферлерін қолданудың мүмкін мүмкіндігі - бұл NETMAP хост стекіне/дан трафикті жіберу үшін жоғары жылдамдықты интерфейспен пайдаланылатын

дестетік сүзгіні енгізу. Бұл жағдайда ОЖ мен қосымшаларға өзгерістер қажет емес және дәстүрлі интерфейстерді қолданады.

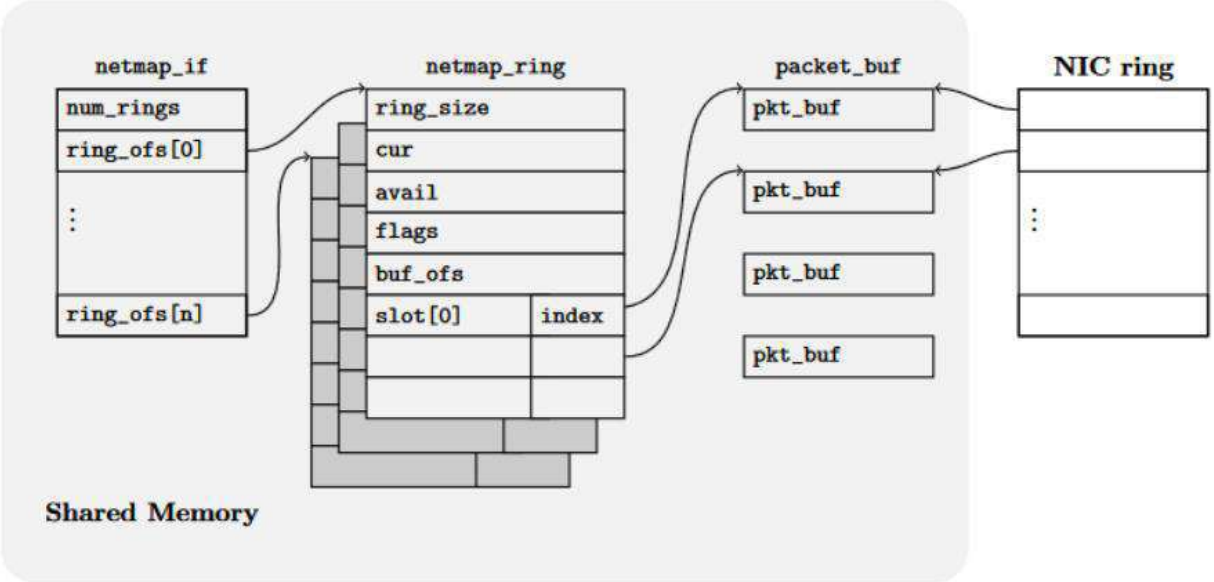
2.1.2 Драйвер

Желілік картаны NETMAP режимінде пайдалану үшін модификацияланған драйверді орнату керек. NETMAP репозиторийінде танымал желілік карта драйверлері үшін пайдалы патчтарды таба аласыз. NVIDIA, Realtek және Intel-дің әртүрлі 1 гигабиттік NIC-ті forcedeth, r8169 және e1000 драйверлері қолдайды. Сонымен қатар, виртуалды интерфейстер үшін virtio, ixgbe және i40e драйверлері сәйкесінше 10 гигабиттік және 40 гигабиттік Intel желілік карталары үшін қол жетімді. Netmap-тің жергілікті драйверлерінің модификациялары тән және өңдеулердің аз санынан тұрады, бұл әртүрлі желілік карталар үшін NETMAP қолдауын жүзеге асыруды жеңілдетеді.

NETMAP қолданбасы белсенді емес болған кезде, желілік карта драйвері әдепкі бойынша жұмыс істейді. Желілік карта операциялық жүйемен басқарылады және дестетерді стандартты интерфейстерге NETMAP қосымшасы іске қосылып, оны басқарғанға дейін жеткізеді.

2.1.3 Сәулеті

2.2-суретте NETMAP қолданылатын негізгі деректер құрылымдары көрсетілген.



2.2 сурет – NETMAP деректер құрылымы

Netmap_if, netmap_rings және packet_bus деректер құрылымдары ядро бөлетін және барлық процестермен бөлісетін бір тұрақты жад бөлімінде орналасқан. Әр процесс өзінің виртуалды мекенжай кеңістігінде жұмыс істейтіндіктен, сілтемелер салыстырмалы түрде сақталады. Бұл салыстырмалы сілтемелер әр процеске өзінің мекен-жай кеңістігіне

қарамастан, деректер құрылымының жадындағы дұрыс позицияны есептеуге мүмкіндік береді. Барлық аталған деректер құрылымдары желілік картаны NETMAP режимінде пайдалану кезінде ерекшеленеді, бұл жұмыс уақытында жадтың кідіруіне жол бермейді.

Netmap_if деректер құрылымында тек оқуға болатын желілік интерфейс туралы ақпарат бар. Netmap_if netmap_rings санын num_rings және ring_of массивін netmap_rings көрсететін элементтермен сақтайды.

Netmap_rings қоңырау буфері кіріс және шығыс трафикті бөлек буферлейтін тәуелсіз желілік интерфейс буферлеріне сілтеме жасайды. Әр сақинада элементтердің максималды саны (ring_size), ағымдағы оқу немесе жазу позициясының көрсеткіші (cur), бос немесе бос емес элементтер саны (avail) және сақина мен packet_buf деректер құрылымының басталуы арасындағы ығысуды қамтитын өріс (buf_ofs) бар. Сонымен қатар, сақинада packet_buffer элементтері бар ring_size элементтері бар slot[] массиві бар.

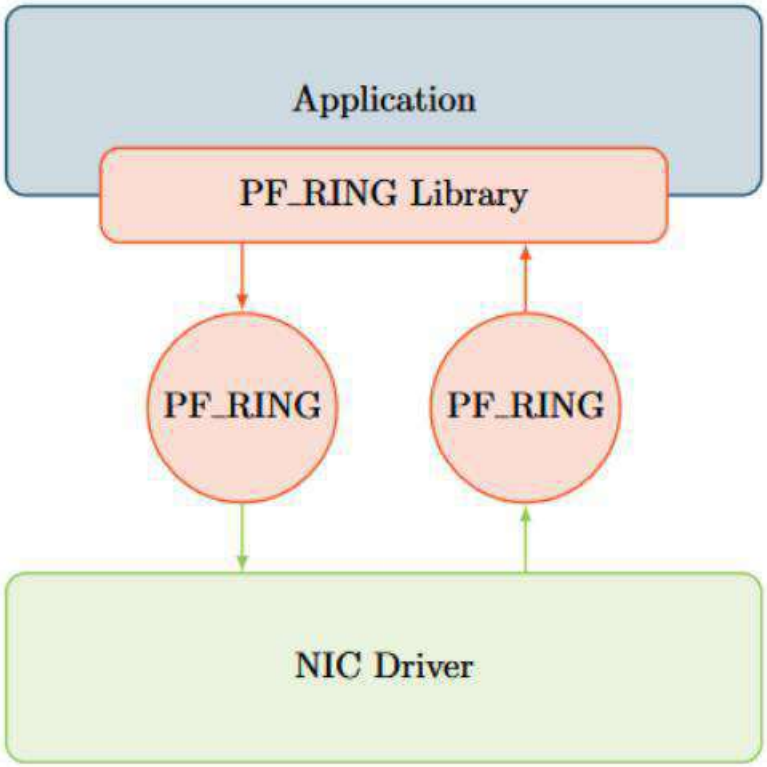
Packet_buf-тің әр элементінде бір желілік кадр бар. Бұл элементтер ядро процестері, пайдаланушы процестері және DMA Engine желілік картасы арасында қол жетімді. Әр ұяшықта айнымалы түрінде әр десте үшін, мысалы үшін және желілік дестетің ұзындығы үшін қосымша ақпарат болады.

2.2 PF_RING

PF_RING - бұл жылдамдықты және сәйкесінше дестетік басып алу тиімділігін арттыратын ашық бастапқы код кітапханасы. PF_RING 2.6.32 және одан жоғары нұсқаның Linux ядросының модулі ретінде де, бөлек фреймворк ретінде де қол жетімді. Кітапхана қосымша компоненттерді пайдалануға мүмкіндік беретін модульдік архитектураға ие: нөлдік көшірме, FPGA негізіндегі желілік карталарды қолдау (Napatech, Accolade, Mellanox, Exablaze, Myricom/CSPI, Endace, Fiberblaze, Inveatech, Netcope вендорлар).

2.2.1 Шолу

PF_RING кітапханасын пайдаланатын типтік қосымшаның өзара әрекеттесу құрылымы 2.3 суретте көрсетілген. Бағдарлама PF_RING дестетік буферіне кіру үшін кітапхананы пайдаланады. Дестетерді жіберу және қабылдау үшін бөлек буферлік сақиналар бар. Желілік карта драйвері дестетерді қабылдау немесе жіберу кезінде бірдей буферлік сақиналарды пайдаланады.



2.3 сурет – PF_RING пайдалану кезінде қосымшаның, ОЖ және желілік картаның өзара әрекеттесу құрылымы

2.2.2 Драйвер

Бір онжылдықта PF_RING-тің әртүрлі нұсқалары жасалды, олардың әрқайсысының өзіндік архитектурасы бар. PF_RING негізгі дестетік буфері тұрақты болып қалғанымен, драйвер мен кітапхананың функциялары мен өнімділігі нұсқадан нұсқаға өзгерді.

PF_RING артықшылықтарының бірі – оны стандартты Linux желілік карта драйверлерімен пайдалану мүмкіндігі, бұл бір жағынан жоғары икемділік береді, ал екінші жағынан кітапхананың өнімділігі төмен. PF_RING нұсқаларының бұл түрі желілік карта мен буферлік сақиналар арасындағы дестетерді алмасу өнімділігін арттыратын in-kernel драйверлеріне жатады.

TNAPI драйверлерінің нұсқалары да болды, олардың қолдауы қазір тоқтатылды. Олар трафикті бақылау үшін арнайы жасалған, сондықтан дестетерді бірнеше өңдеу ағындары бойынша қабылдау мен таратуды ғана қолдады.

Ең жоғары өнімділікке kernel-bypass деп аталатын драйверлерді пайдалану арқылы қол жеткізіледі, олар дестетерді тікелей пайдаланушы кеңістігіне және одан операциялық жүйенің өзегін айналып өтіп, дестетерді көшірмей жібереді. PF_RING-тің бұл нұсқасы DNA деп аталды және Libzero деп аталатын кітапхана болды, олар дестетерді өңдеуге ыңғайлы мүмкіндіктерді ұсынады.

PF_RING DNA-ның Libzero-мен жалғасы PF_RING ZC (Zero Copy) болды, ол өзінің алдындағы барлық артықшылықтарды мұра етті және бүгінгі күнге дейін дамып келеді.

PF_RING ZC in-kernel режимінде де, kernel-bypass режимінде де қолданыла алады. PF_RING ZC нұсқасы жеке меншік болып табылады және Intel желілік карталарына арналған модификацияланған драйверлерді қамтиды (e1000e, igb, ixgbe, i40e, fm10k).

2.2.3 Сәулеті

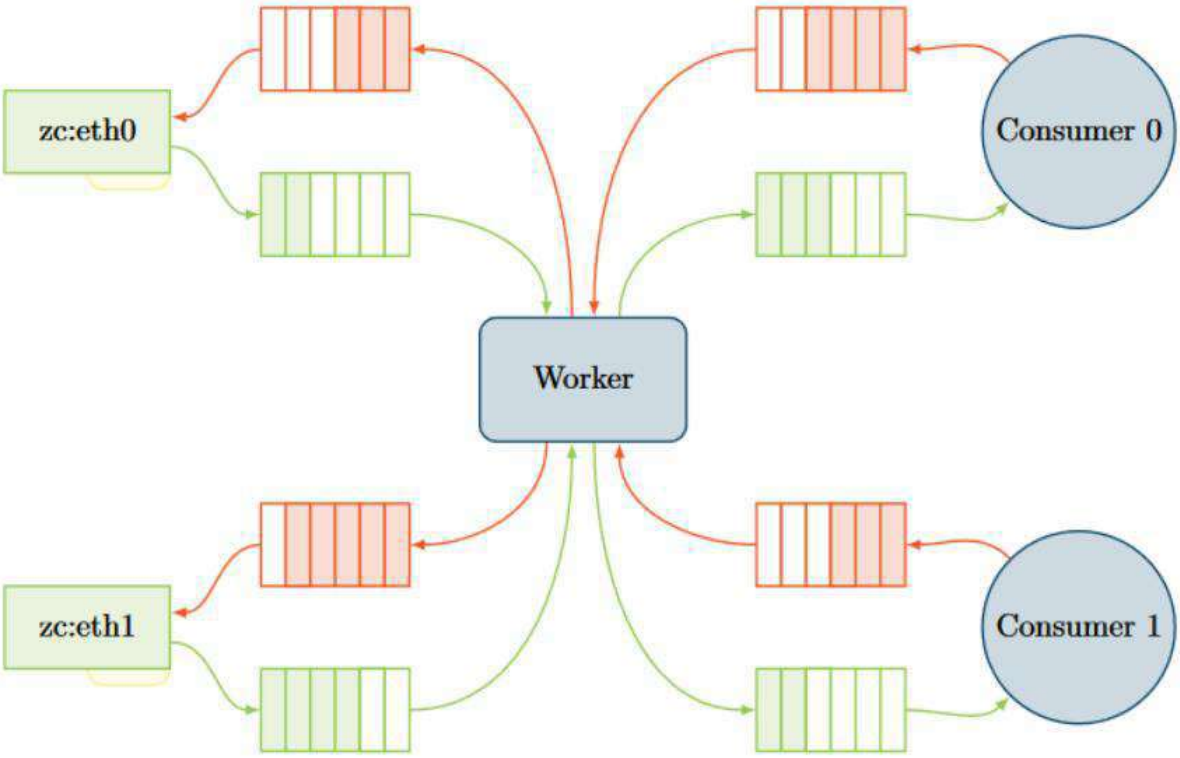
PF_RING ZC кітапханасының бастапқы коды жабық, сондықтан біз архитектураға үстірт шолу жасаймыз.

PF_RING ZC қосымшаларды оңай іске асыру үшін негізгі құрылыс блоктарының бір түрін ұсынады. Кітапхананың маңызды ерекшеліктерінің бірі - дестетерді кезектер, ағындар мен процестер арасында көшірусіз беру.

Тағы бір маңызды құрылым – pfring_zc_cluster, ол ID арқылы анықталады және олардың арасында дестетерді бөлісу үшін ағындарды немесе процестерді топтастыру үшін қолданылады.

Дестетерді өңдеу нысандары (желілік карталар, процестер немесе ағындар) арасында беру үшін pfring_zc_queue қолданылады. Кезектерді pfring_zc_multi_queue-ге қосуға болады.

pfring_zc_worker дестетерді әртүрлі нысандар арасында тарату үшін қолданылады. Теңгергіш және сплиттер (fanout) екі түрлі pfring_zc_worker негізінде жасалады. Теңгеруші бір немесе бірнеше кіріс кезектерден келетін дестетерді бір немесе бірнеше шығыс кезектерге бөле алады. Бөлу функциясын бағдарламалық түрде өзгертуге болады. Сплиттер кіріс дестетерді тарату үшін бірнеше шығыс кезегін қолданады. pfring_zc_buffer_pool-бұл pfring_zc_worker-ам сияқты қолданылатын жад бассейні. 2.4-суретте PF_RING ZC cluster көмегімен қосымшаның құрылымы көрсетілген. Ол екі желілік картадан, дестетерді таратуға арналған бір worker және екі дестетік "тұтынушылардан" (consumers) тұрады. Желілік карталар, жұмысшы және "тұтынушылар" дестетерді қабылдау және беру үшін жеке pfring_zc_queues кезектерін пайдаланады. Worker желілік трафикті теңгеруші ретінде де, pfring_zc_multi_queue көмегімен желілік дестетерді әр "тұтынушыға" жеткізуге арналған сплиттер ретінде жүзеге асырылуы мүмкін. "Тұтынушылар" ағындар немесе жеке процестер, өңдеу дестетері сияқты жүзеге асырылуы мүмкін. Бұл мысалда дестетер worker-ге қайтарылады, басқа шешімдер ретінде сіз дестетерді басқа өңдеушіге апаратын кезекке бағыттай аласыз немесе тек дестетерді өңдей аласыз.



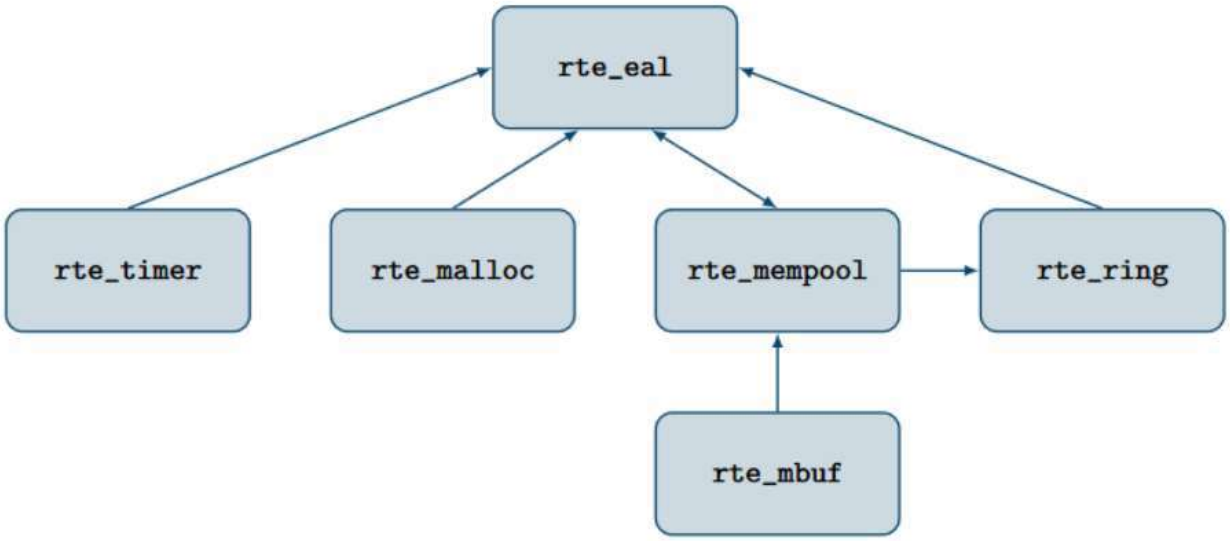
2.4 сурет – PF_RING ZC Cluster

2.3 DPDK

Intel компаниясының open source фреймворкын қарастырыңыз – Data Plane Development Kit (DPDK). DPDK - бұл желілік дестетерді жылдам өңдеуге арналған кітапханалар мен драйверлерді жинау.

2.3.1 Шолу

Жақтаудың негізгі мақсаты тек дестетерді енгізу/шығару механизмдерін жеделдету емес - DPDK дестетерді кешенді өңдеуге арналған кітапханалар мен компоненттер жиынтығын ұсынады. DPDK негізіндегі қосымшада өңдеу тапсырмалары үшін қажетті белгілі бір кітапханаларды таңдау мүмкіндігі бар. 2.5-суретте олардың негізгілері көрсетілген. Барлық кітапханалар rte_real (EAL, Environment Abstraction Layer) деп аталады. EAL дестетерді өңдеудің әртүрлі ағындарын іске қосады және басқарады.



2.5 сурет – DPD құрамындағы негізгі кітапханалар

2.3.2 Драйвер

Бұл қаншалықты айқын көрінсе де, Intel 1 Gbit, 10 Gbit және 40 Gbit желілік карталарына арналған драйверлер DPDK құрамына кіреді. Сонымен қатар, виртуализацияда қолданылатын бірнеше драйверлер DPDK-ге бейімделген. Барлық драйверлер Poll Mode Driver (PPD) деп аталады, яғни.желілік картадағы дестетердің бар-жоғын кез-келген үзілістерді пайдаланбай білуге болады. PMD пайдаланған кезде желілік карталар осы драйверге қосылуы керек. PMD драйвері желілік құрылғыны амалдық жүйеден ажыратады. Желілік картаны PMD-ге/ден байланыстыруға/шешуге арналған арнайы сценарий бар pci_unbind.py.

PMD драйвері igb_uio драйверіне негізделген. UIO драйверлері арнайы кеңістіктегі дестетік өңдеудің көптеген әрекеттерін орындау үшін арнайы жасалған. Linux ядро модулі түрінде UIO драйверін енгізу бар, бірақ оның мүмкіндіктері шектеулі.

2.3.3 Сәлеті

Rte_eal негізгі кітапханасы аппараттық құралдар мен ОЖ-ны іске асырудың егжей-тегжейін ескере отырып, қосымшаларға арналған интерфейсті ұсынады. Сонымен қатар, EAL бағдарлама мен ресурстарды (жад, PCI компоненттері, таймерлер және т.б.) баптандырады.

Time Manager (rte_timer) функцияларды асинхронды, бір рет немесе мезгіл-мезгіл шақыруға мүмкіндік береді. Анықтамалық дәлдік EAL қамтамасыз етеді.

Жад менеджері (rte_malloc) компоненті жадты тиімді пайдалану үшін huge pages-те жадты бөледі.

Network Packet Buffer Management (rte_mbuf) дестетік буферлерді жасай немесе жоя алады. Бұл буферлерді кез-келген деректерді сақтау үшін

пайдалануға болады. Бағдарлама іске қосылған кезде барлық буферлер жасалады.

Ring Manager (rte_ring) бекітілген өлшемдегі бос сақиналы буферлерді құлыптауды сұрайды. Бұл сақиналар бірнеше "өндірушілер" мен "тұтынушыларды" қолдайды. Сақиналарды әртүрлі ағындар арасындағы байланыс үшін пайдалануға болады.

Жад бассейн менеджері (rte_mempool) объектілерді сақиналарда сақтау үшін қолданылады. Сонымен қатар, әр CPU өзегі үшін буфер бар. Нысандар жадқа қол жеткізуді оңтайландыру үшін қол жетімді жад арналары арқылы біркелкі бөлінеді.

DPDK сонымен қатар жоғары деңгейлі функционалдылық үшін кітапханаларды ұсынады, мысалы liberty_hash кітапханасы хэш негізіндегі дестетерді қайта бағыттау үшін қолданылады немесе librte_lpm ұзақ префикстерді сәйкестендіру алгоритмін ұсынады. Librte_lpm кітапханасы IP, TCP, UDP және SCTP сияқты протоколдарды өңдеудің әдістерін ұсынады.

3 Дестелерді басып алу өнімділігі мен трафикті жіктеуді тестілеу

Желіні жобалау және басқару желілік инфрақұрылымды да, қосылатын құрылғылардың жұмысын да, әр желілік құрылғының дестетерін өңдеу тәсілдерін де терең түсінуді қажет етеді. Әдетте, желілік құрылғылардың өнімділігі секундына битпен көрсетілген интерфейстердің жылдамдығын білдіреді (бит/с, bps). Мысалы, желілік құрылғыны секундына 10 гигабит өнімділігі бар деп сипаттауға болады (Гбит/с, Gbps). Бұл пайдалы және маңызды ақпарат болғанымен, секундына тек биттер тұрғысынан өнімділікті білдіру желілік құрылғылардың басқа да маңызды көрсеткіштерін объективті түрде қамтымайды. Сондықтан, әртүрлі Ethernet дестетерінің өлшемдері үшін тиімді беру және деректерді өңдеу жылдамдығын анықтау маңызды ақпаратты және желінің және/немесе құрылғының сипаттамаларын толық түсінуді қамтамасыз етеді.

Бұл тарауда соңғы түйінде әртүрлі мөлшердегі Ethernet рамаларын өңдеу өнімділігін тексеру нәтижелері келтірілген және қарастырылған, сонымен қатар трафикті жіктеудің бірнеше кітапханалары сыналады.

3.1 Сынақ стендінің сипаттамасы

2 Nutanix NX-3050 сервері, олардың әрқайсысы 3.1-кестеде келтірілген сипаттамаларға ие.

3.1 кесте – Nutanix NX-3050 серверінің сипаттамалары

Атауы	Сипаттама
Процессор	2 x Intel Xeon E5-2670 8 cores / 16 threads / 2.6 GHz Base|3.3 GHz Turbo
Қойма	2 x 400 GB SSD, 4 x 1 TB HDD
RAM жады	2 x 64 GB (8 x 16 GB DDR3 1600 MHz)
Желілік құрылғылар	2 x 10 GbE, 2 x 1 GbE, 1 x 10/100 BASE-T RJ45
Өлшемдері	Биіктігі: 88mm, Ені: 438mm, Тереңдігі: 679mm
Салмағы	35kg орамасы бар / 30.5 kg орамасыз / 3.2kg нода
Салқындату	4x80mm ШИМ айналу жылдамдығын басқаратын қуатты желдеткіштер
Қуатты тұтыну	1350W максималды, 1100W орта есеппен
Қуат көзі	2 x 1620W Out @180-240V, 10.5-8.0A, 50-60Hz
Жылу бөлу	4610 BTU/hr максималды, 3750 BTU/hr орташа

Әр серверде 10 GbE Intel 82599ES және 1 GbE Intel I350 бар. 10 GbE желілік карталары Cisco SFP-H10GBCU1M Compatible 10G SFP + Passive Direct Attach Copper Twinax кабелімен бір SFP+ трансиверді қолдана отырып тікелей қосылған, 1 GbE желілік карталары қосқышқа екі Cat5E UTP кабельдері арқылы қосылған.

Әрбір сервер CentOS 7 Linux version 3.10.0-327 live-usb кескінінен жүктелген.el7.Соңғы жаңартулар орнатылған x86_64, "Development Tools" дестетер тобы, Enterprise Linux (EPEL) үшін Extra Packag es дестетер

репозиторийі және келесі дестетер: boost-devel kernel-devel-3.10.0-327.el7.x86_64 libpcap-devel numactl-devel GeoIP-devel ccache libcli-devel libnet-devel libnetfilter_conntrack-devel libnl3-devel nacl-devel userspace-rcu-devel zlib-devel ncurses-devel json-c-devel iptables-devel perf htop trace-cmd wget lspci. Барлық орнатылған дестетер фреймворктарды құрастыру және тестілеу кезінде белгілі бір статистиканы алу үшін қажет.

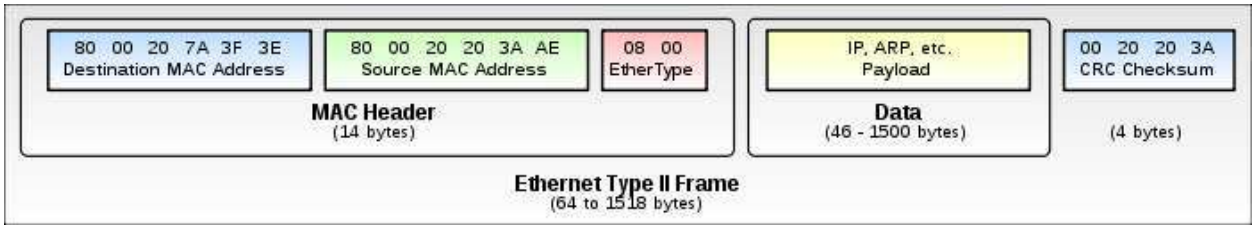
10 GbE Intel 82599ES желілік картасы үшін барлық жағдайларда ixgbe драйвері 4.0.1-k-rh7.2 нұсқасын қолданды, алдын-ала келісілгендерден басқа. 1 GbE желілік карталарында 192.168.1.160/24 және 192.168.1.161/24 мекенжайлары бар eth0 интерфейсі болды, олар арқылы хосттар SSH арқылы қол жетімді болды. 10 GbE желілік карталарында бірінші белсенді enp4s0f0 және enp4s0f1 екі интерфейсі болды.

Бірінші сервер трафик генераторы немесе жіберуші тарап (бұдан әрі – бірінші сервер), екіншісі-қабылдаушы тарап (бұдан әрі-екінші сервер) рөлінде әрекет етеді. Трафик enp4s0f0 интерфейсінде жасалады және қабылданады. Екі серверде де жүйе жүктелгеннен кейін екі NUMA node үшін әр 1 ГБ өлшемі 16 hugepages бөлінді, iptables тізбектері тазартылды, netfilter модульдері жүктелді, SELinux өшірілді.

Трафик генераторы ретінде netsniff-ng trafgen және netmap-тен pkt-gen утилиталары қолданылды.

3.2 Linux стандартты құралдарын тестілеу

10 гигабиттік Ethernet арнасына сәйкес келетін ең аз мөлшердегі секундына дестелердің максималды санын есептейміз. Ethernet жақтауының минималды өлшемі - 64 байт (3.1-суретті қараңыз), оның ішінде 14 байт Ethernet Header алады (6 байт MAC көзі мен тағайындалған мекен-жайы және 2 байт EtherType), 4 байт CRC және 46 пайдалы жүктеме.



3.1 сурет – Ethernet Frame

Жақтауға қосымша InterPacket gap (IPG, 12 байт), MAC Preamble (7 байт) және START frame delimiter (SFD, 1 байт) қосылады. Ия, 802.3 af стандарты үшін JPG мөлшерін 5 байтқа дейін азайтуға болады, бірақ барлық желілік адаптерлер бұл функцияны қолдамайды, сондықтан бастапқы мәнге тоқталайық.

Барлығы, барлығы бір Ethernet фреймінің берілуіне 84 байт алынады. Содан кейін уақыт бірлігіне осындай кадрлардың саны тең (формула 3.1):

$$\frac{10\frac{Gb}{s}}{84\,B} = \frac{10 \cdot 10^9}{84 \cdot 8} = 14\,880\,952.381 \approx 14\,880\,952 \text{ pps} \qquad (3.1)$$

Пайдалы деректердің максималды мөлшері - 1500 байт (MTU), яғни уақыт бірлігіне келетін жақтаулардың саны бір Ethernet кадрына 1538 байт (3.2 формула):

$$\frac{10\frac{Gb}{s}}{1538\,B} = \frac{10 \cdot 10^9}{1538 \cdot 8} = 812\,743.823 \approx 812\,743 \text{ pps} \qquad (3.2)$$

Біз бір дестені өңдеуге жұмсалған CPU циклдерінің санын өрескел есептейміз, бір цикл үшін бір нұсқаулықтың орындалуын және суперскалярлық процессорларды ескермейміз. Мұны істеу үшін алдымен минималды өлшемдегі бір Ethernet фреймін өңдеу қажет болатын максималды уақытты табыңыз:

$$\frac{1}{14\,880\,952\ packets/s} = 67.2 \cdot 10 - 9\text{s} = 67.2 \text{ ns} \qquad (3.3)$$

Содан кейін 1 CPU 3.3 GHz үшін CPU тактыларының саны:

$$3.3 \text{ GHz} \cdot 62.7 \text{ ns} = 206.91 \approx 207 \text{ CPU cycles}$$

Яғни бізде ең аз мөлшердегі Ethernet фреймін өңдеуге арналған 207 CPU бар. Біз бұл сандарды нақты нәрсемен салыстырамыз:

- кэш қателіктері (cache misses).

Бір кэшті жіберіп алу 32 ns жұмсайды, яғни екі кэшпен 64 ns жұмсалады. Linux sk_buff 4 кэш жолын алады, skb_buff үлестіру кезінде ядро бұл жолдарды нөлдермен толтырады.

- кэшті тарату (cache references).

Кейде біз L2 және L3 кэшіне (cache hit) қол жеткізе аламыз, содан кейін Intel Xeon E5-2670 процессорының L2 және L3 кэштеріне кіру уақыты сәйкесінше 3.5 ns және 6.5 ns құрайды.

- бұғаттау операциялары.

Құрастыру нұсқауларында "LOCK" (құлыптау) префиксі болуы мүмкін, бұл олардың атомдық операцияны орындауын білдіреді. Бұл префикс, мысалы, спинлокты (spinlock) жасау немесе босату кезінде қолданылады. Қарастырылып отырған процессор үшін атомдық құлыптау операциясы 8.25 ns алады, айналдыру тек бір CPU-ға тиген жағдайда екі құлыптау қоңырауы болады, яғни 16.5 ns.

- жүйелік қоңыраулар.

Мысал ретінде, Linux ядросында getuid жүйелік қоңырауы 87.77 ns (немесе CONFIG_AUDITSYSCALL ядросының өшірілген нұсқасы бар 41.85 ns) және CPU тиімділігі 1.42-ге тең 201 CPU сағатты алады, яғни жүйелік

қоңыраудың өзі дестетің максималды өңдеу уақытына қарағанда көп уақытты алады.

3.2.1 Бастау үшін, желілік стекке қатысты параметрлер әдепкі бойынша орнатылған, жоспарланбаған жүйеде дестетерді басып алу өнімділігін тексереміз. Бірінші сервердің желілік адаптерінің драйверінде әдепкі бойынша параметрлері бар, RSS=0, RX/TX checksumming on, scatter-gather on, TSO on, UFO off, GSO on, GRO on, LRO on, 32 RX rings, RX/TX ring buffer size 512. Біз irqbalance және tuned өшірмейміз.

Irqbalance өнімділікті арттыру үшін аппараттық үзілістерді мультипроцессорлық жүйелердегі өзектерге таратады. Tuned - өнімділікті арттыру немесе баяулату үшін жүйенің параметрлерін динамикалық түрде өзгерту үшін жүйелік ресурстарды пайдалану туралы ақпарат жинайтын демон.

Екінші сервердің желілік адаптері promiscuous mode-ға ауыстырылды:

ip link set enp4s0f0 promisc on

Бұл режимде желілік карта кімге бағытталғанына қарамастан барлық дестетерді қабылдай бастайды.

Сондай-ақ, екінші сервердің желілік картасында Ethernet Flow Control өшірілген:

ethtool -A enp4s0f0 rx off tx off autoneg off

Бұл жағдайда желілік адаптер pause фреймы жібермейді, осылайша шығындарды болдырмау үшін кіретін дестетерді өңдей алмайтындығы туралы хабарлайды.

Біз қабылданған дестетердің статистикасын sysfs (/sys/class/<iface>/statistics/rx_packets) көмегімен 1 секундтық аралықпен түсіреміз, яғни қабылданған дестетерді пайдаланушы кеңістігіне көшіруді алып тастаймыз және оларды Linux желілік стек деңгейінде тастаймыз. Қорытынды нәтиже 10 өлшеудің орташа санымен алынды.

Бірінші серверде trafgen утилитасы арқылы UDP және TCP өлшемді дестетер жасалды 46, 86, 186, 486, 986, 1486 байт (64, 104, 204, 504, 1004, 1504 байт Ethernet фреймдер тиісінше) секундына дестетердің максималды тиісті жылдамдығымен. Пайдалы жүктеме (payload) нөлдермен толтырылды.

Өлшеу нәтижелері 3.1 және 3.2 кестелерде жинақталған, 3.2 және 3.3-суреттерде орташа мәндер бойынша тиісті графиктер салынған.

3.1 кесте – UDP дестетерін басып алу өнімділігі

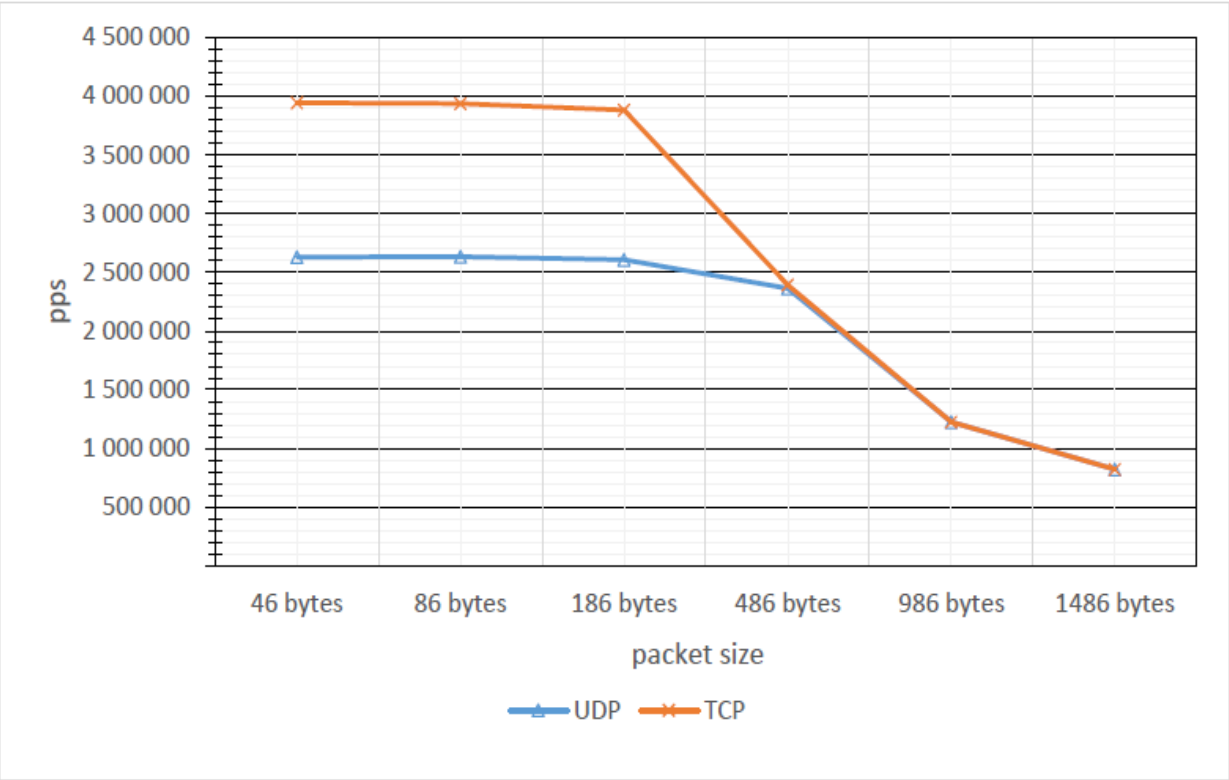
№	UDP 46 bytes	UDP 86 bytes	UDP 186 bytes	UDP 486 bytes	UDP 986 bytes	UDP 1486 bytes
1	2 627 392	2 632 704	2 606 272	2 363 456	1 223 539	822 247
2	2 627 264	2 632 768	2 606 720	2 363 648	1 223 698	822 356

3.1 кестенің жалғасы

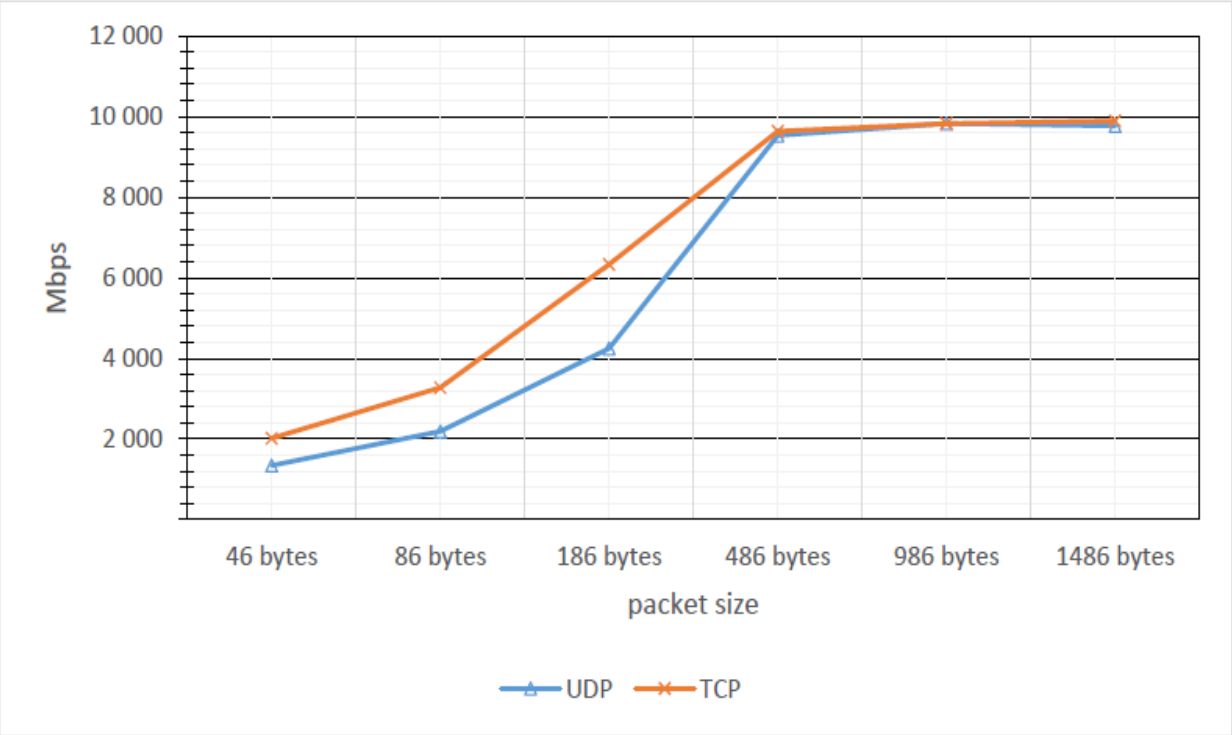
№	UDP 46 bytes	UDP 86 bytes	UDP 186 bytes	UDP 486 bytes	UDP 986 bytes	UDP 1486 bytes
3	2 627 072	2 632 448	2 605 888	2 364 032	1 223 473	822 418
4	2 627 137	2 632 448	2 606 528	2 364 032	1 223 374	822 182
5	2 627 584	2 631 360	2 605 760	1 364 288	1 223 484	822 251
6	2 629 057	2 632 256	2 606 080	2 364 096	1 223 485	822 139
7	2 628 928	2 632 448	2 606 144	2 363 392	1 223 484	822 165
8	2 629 504	2 632 256	2 606 144	2 363 712	1 223 471	822 151
9	2 629 184	2 632 384	2 606 144	2 364 160	1 223 512	822 281
10	2 629 440	2 632 320	2 606 080	2 363 968	1 223 458	822 174
Avg.pps	2 628 256	2 632 339	2 606 176	2 363 878	1 223 498	822 236
Avg.Mbps	1 345 667	2 109 106	4 253 279	9 531 156	9 827 136	9 774 742

3.2 кесте – TCP дестетерін басып алу өнімділігі

№	UDP 46 bytes	UDP 86 bytes	UDP 186 bytes	UDP 486 bytes	UDP 986 bytes	UDP 1486 bytes
1	3 947 264	3 936 960	3 883 776	2 390 893	1 223 668	822 154
2	3 943 040	3 940 992	3 882 880	2 390 764	1 223 589	822 139
3	3 943 616	3 936 832	3 878 400	2 390 752	1 223 642	822 095
4	3 940 928	3 937 664	3 879 744	2 390 752	1 223 477	822 161
5	3 943 040	3 937 152	3 881 088	2 390 838	1 223 492	822 304
6	3 943 808	3 937 856	3 881 280	2 390 716	1 223 597	822 126
7	3 942 848	3 937 152	3 879 749	2 391 273	1 223 593	822 160
8	3 943 872	3 937 280	3 881 355	2 390 967	1 223 650	822 229
9	3 943 872	3 938 368	3 881 240	2 390 956	1 223 944	822 263
10	3 943 360	3 937 408	3 880 180	2 390 910	1 223 526	822 094
Avg.pps	3 943 488	3 937 766	3 880 969	2 390 882	1 223 618	822 173
Avg.Mbps	2 019 066	3 276 221	6 333 741	9 640 036	9 828 099	9 892 386



3.2 сурет – UDP және TCP дестетерінің орташа өнімділікті басып алу графиктері



3.3 сурет – Өңделетін трафик көлемінің орташа мәндерінің графигі

3.2.2 Енді Linux ядросы мен желілік адаптердің кейбір параметрлерін өзгертіңіз. RX және TX сақиналы буферлердің мөлшерін көбейтіңіз:

ethtool -G enp4s0f0 rx 4096 tx 4096

Біз сәйкесінше UDP және TCP үшін трафик ағындарын жіктеу ережелерін орнатамыз:

ethtool -N enp4s0f0 rx-flow-hash udp4 sdfn

ethtool -N enp4s0f0 rx-flow-hash tcp4 sdfn

Мұндағы udp4 - UDP IPv4, tcp4 - TCP IPv4, sdfn - бастапқы (s) және тағайындалған (d) IP мекен-жайының хэшін есептеуде 4 (f) деңгей тақырыбының 0 және 1 байттарын, келген дестетің 2 және 3 байттарын (n) пайдалануды білдіреді.

Irqbalance өшіріңіз және арнайы сценарийді қолданыңыз set_irq_affinity.sh ixgbe драйверімен бірге жеткізілген, желілік картаның үзілістерін процессордың барлық өзектеріне теңестіру үшін. Tuned-де біз throughput-performance профилін орнатамыз.

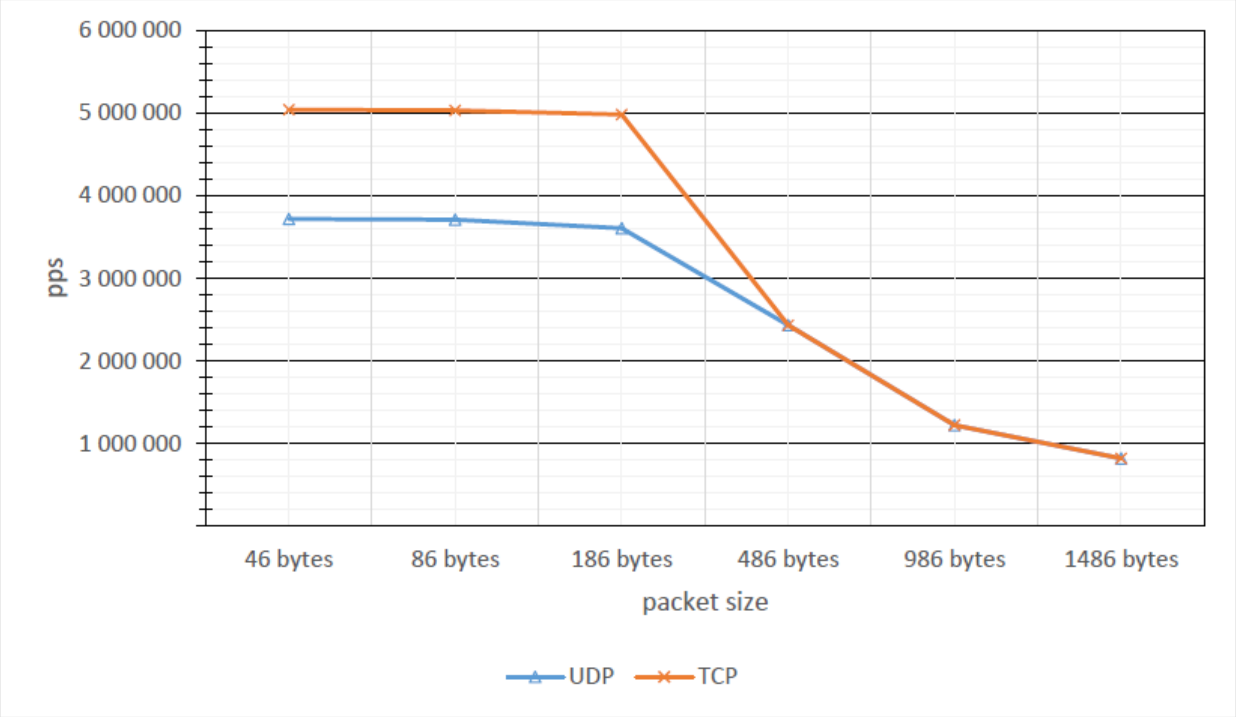
Сол сияқты, бірінші серверде trafgen утилитасын қолдана отырып, UDP және TCP өлшемді дестетер жасалды 46, 86, 186, 486, 986, 1486 секундына дестетердің максималды жылдамдығы бар байт. Пайдалы жүктеме (payload) нөлдермен толтырылды. Өлшеу нәтижелері 3.3 және 3.4 кестелерде жинақталған, 3.4 және 3.5-суреттерде орташа мәндер бойынша тиісті графиктер салынған.

3.3 кесте – UDP дестетерін басып алу өнімділігі

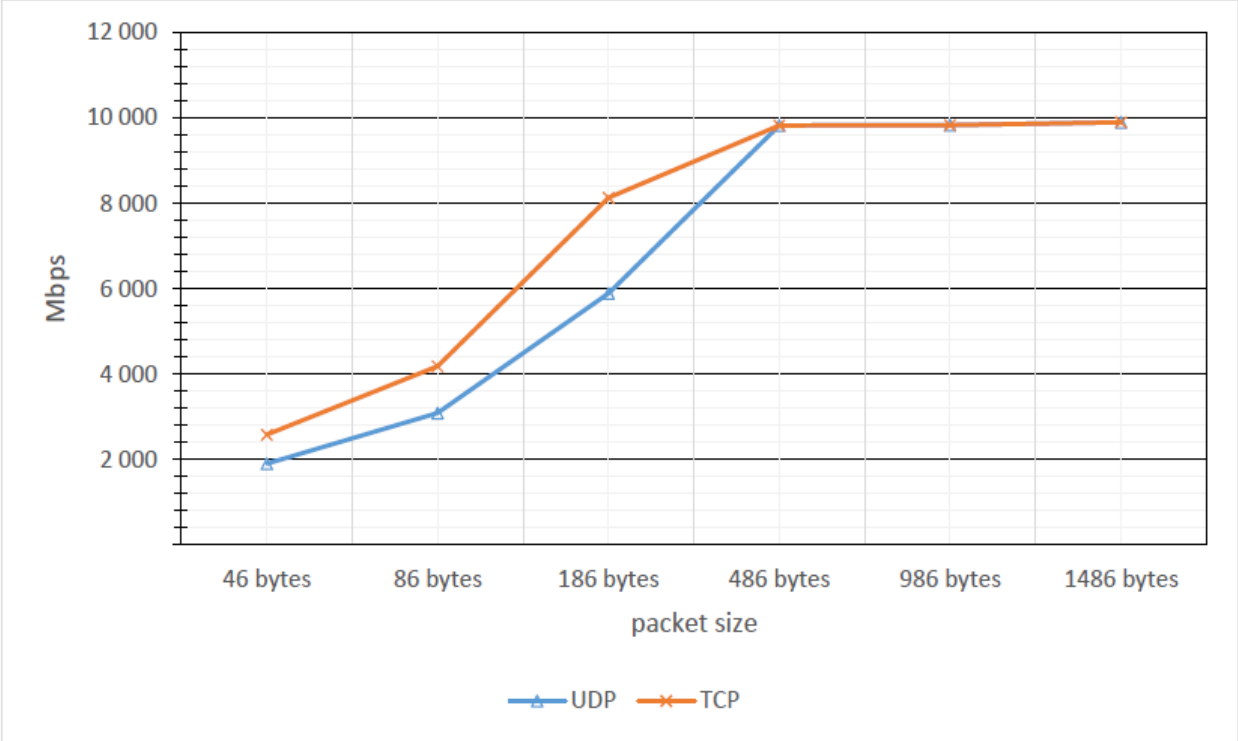
№	UDP 46 bytes	UDP 86 bytes	UDP 186 bytes	UDP 486 bytes	UDP 986 bytes	UDP 1486 bytes
1	3 723 200	3 712 234	3 610 752	2 435 956	1 223 548	822 148
2	3 724 032	3 713 652	3 610 240	2 435 155	1 223 433	822 153
3	3 720 512	3 710 410	3 609 920	2 343 842	1 223 852	822 110
4	3 721 664	3 711 740	3 615 296	2 433 910	1 223 322	822 272
5	3 721 536	3 711 365	3 609 856	2 434 950	1 223 584	822 298
6	3 716 096	3 714 188	3 610 240	2 435 014	1 223 355	822 257
7	3 721 216	3 710 475	3 609 856	2 435 845	1 223 450	822 180
8	3 719 360	3 713 589	3 651 840	2 434 452	1 223 511	822 179
9	3 720 704	3 712 495	3 581 760	2 434 842	1 223 474	822 164
10	3 720 896	3 712 320	3 581 696	2 433 473	1 223 342	822 166
Avg.pps	3 720 922	3 712 247	3 609 146	2 434 871	1 223 487	822 193
Avg.Mbps	1 905 112	3 088 589	5 890 126	9 817 399	9 827 048	9 892 626

3.4 кесте – TCP дестетерін басып алу өнімділігі

№	UDP 46 bytes	UDP 86 bytes	UDP 186 bytes	UDP 486 bytes	UDP 986 bytes	UDP 1486 bytes
1	5 043 154	5 029 821	4 974 682	2 435 485	1 223 770	822 085
2	5 039 856	5 032 182	4 982 752	2 435 125	1 223 851	822 154
3	5 042 120	5 033 352	4 983 352	2 435 652	1 223 187	822 198
4	5 040 421	5 031 175	4 979 285	2 435 452	1 223 358	822 098
5	5 041 072	5 032 090	4 981 276	2 435 735	1 223 412	822 257
6	5 039 965	5 032 845	4 979 653	2 435 658	1 223 922	822 161
7	5 042 423	5 032 963	4 982 397	2 436 120	1 223 452	822 180
8	5 042 647	5 032 423	4 982 752	2 435 954	1 223 351	822 296
9	5 041 852	5 033 142	4 982 940	2 435 847	1 223 285	822 223
10	5 043 122	5 032 635	4 983 311	2 435 816	1 223 468	822 162
Avg.pps	5 041 663	5 032 263	4 981 240	2 435 684	1 223 506	822 181
Avg.Mbps	2 581 331	4 186 843	8 129 384	9 820 678	9 827 200	9 892 482



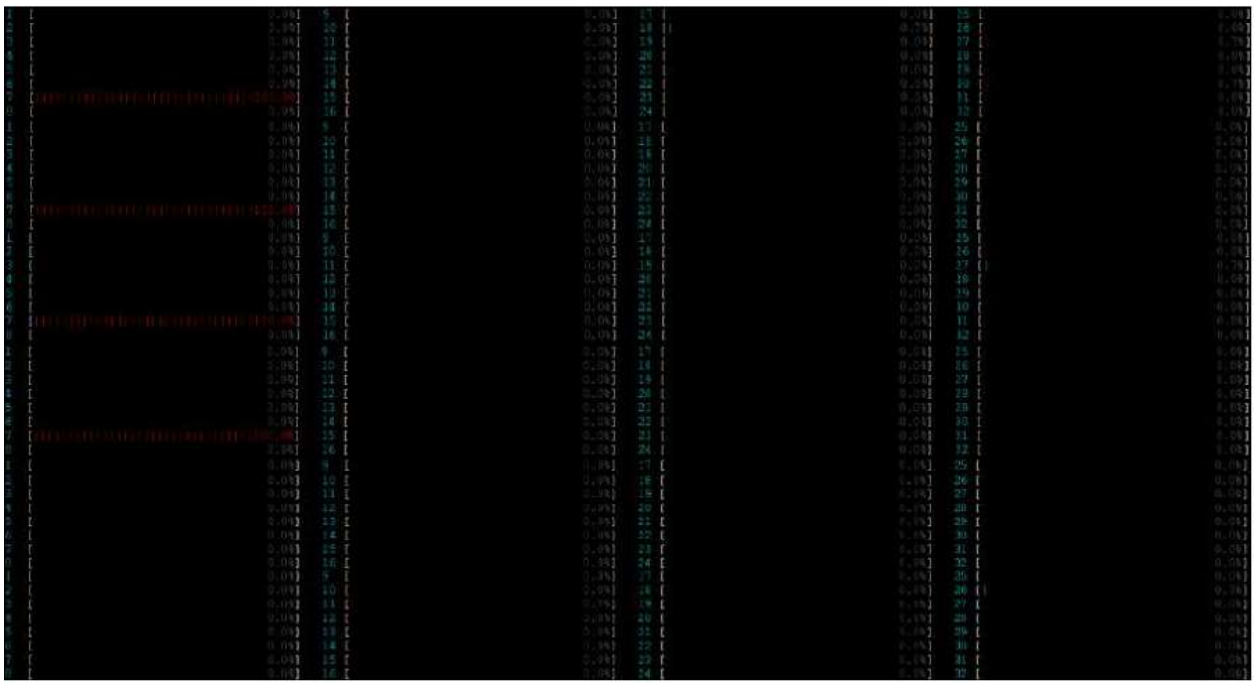
3.4 сурет – UDP және TCP дестетерінің орташа өнімділікті басып алу графиктері



3.5 сурет – Өңделетін трафик көлемінің орташа мәндерінің графигі

486, 986, 1486 байт өлшемді UDP және TCP дестетері барлық 10 гигабиттік арнаны оңай толтырады және ішінара Linux желілік стекімен өңделеді.

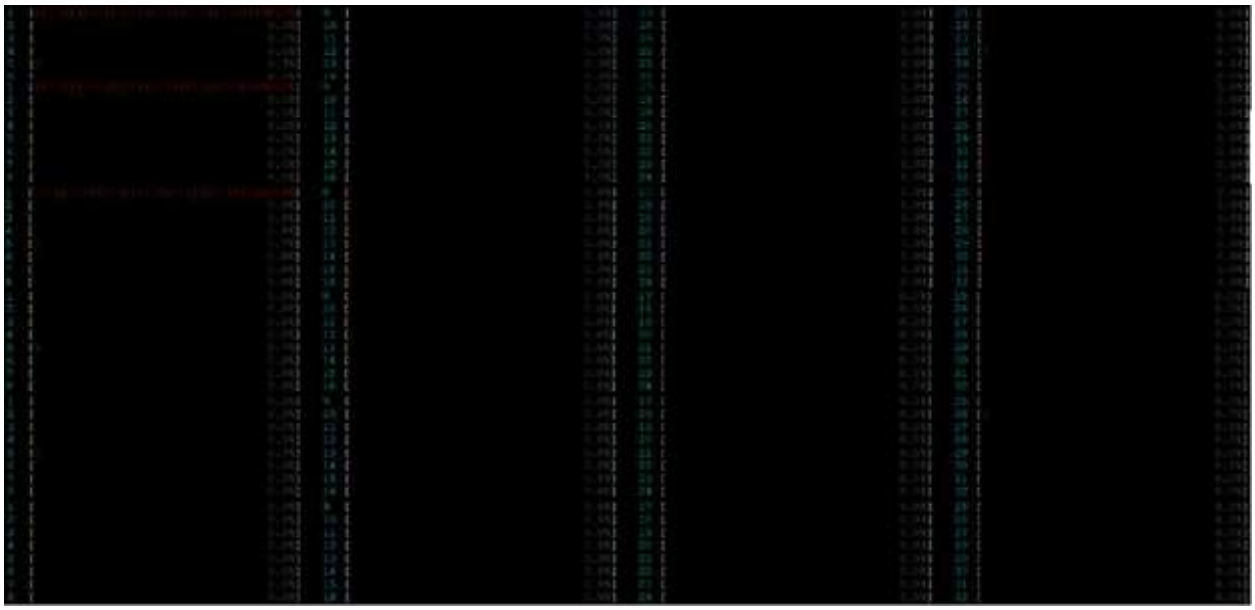
3.2 және 3.4 суреттеріне сүйене отырып, ядро мен желілік картаның кейбір параметрлерін түзетіп, Linux желілік стекінде UDP және TCP дестетерін шағын көлемде (46, 86, 186) басып алу өнімділігі артты, алайда RSS-ті қолдануға және процессордың барлық ядроларына үзілістерді таратуға қарамастан, кішкентай дестетерде тек бір ядро жүктелген (3.6 және 3.7 суреттерді қараңыз). Мұны түсіндіруге болады, іс жүзінде белгілі бір типтегі трафиктің бір ағымы болды, өйткені бірдей дестетер үнемі пайда болды. Мұны растау үшін тест жүргізілді: бірінші серверде trafgen көмегімен кездейсоқ IP және Port көзі бар 54 байттық TCP SYN дестетері жасалды. Тест нәтижелері 3.5-кестеге енгізілген.



3.6 сурет – UDP дестетерін басып алу тексеру кезінде htop екінші сервердің шығару бөлігі (жоғарыдан төменге қарай 46-1486 байт)

3.5 кесте – TCP SYN дестетерін басып алу өнімділігі

№	TCP SYN 54 bytes
1	10 175 425
2	10 168 284
3	10 164 745
4	10 171 254
5	10 207 285
6	10 174 541
7	10 172 470
8	10 173 000
9	10 169 827
10	10 181 051
Avg. pps	10 175 788



3.7 сурет – TCP дестетерін басып алу тексеру кезінде htop екінші сервердің шығару бөлігі (жоғарыдан төменге қарай 46-1486 байт)

Бұл жағдайда процессордың өзектерін жүктеу минималды деңгейде болды (3.8-суретті қараңыз). CPU-дің төмен жүктелуі бақылау сомасын есептеуді желі адаптеріне, RSS-тің сәтті жұмысына, GRO-ның NAPI-мен тиімді әрекеттесуіне және дестетерді пайдаланушы кеңістігіне көшірудің болмауына байланысты болуы мүмкін.

3.8 сурет – TCP SYN дестетерін басып алуды тексеру кезінде екінші сервердің htop шығару бөлігі

Салыстыру үшін, Linux желілік стекіндегі дестетердің қосымша көшірмесі қаншалықты әсер етеді 3.6-кестеде пайдаланушы қосымшасының розеткасына келетін және бір ядромен өңделетін бірдей UDP 46 байт дестетерінің максималды мөлшерін өлшеу нәтижелері келтірілген. Ол үшін recvmmsg() жүйелік қоңырауына негізделген сценарий қолданылды. Recvmsg() жүйелік қоңырауы бір қоңырау үшін розеткадан бір уақытта бірнеше дестеті алуға мүмкіндік береді.

3.6 кесте – UDP сокеттер розеткасына келетін дестетер саны

№	UDP 46 bytes
1	1 013 852
2	994 384
3	999 685
4	1 009 181

3.6 кестенің жалғасы

№	UDP 46 bytes
5	1 009 352
6	1 005 188
7	1 009 285
8	1 022 771
9	1 044 137
10	1 039 685
Avg. pps	1 014 752
Avg. Mbps	519 553

Linux желілік стекінде не болып жатқанын түсіну үшін UDP дестеін алған кезде біз trace-cmd (ftrace) утилитасын қолданамыз. Сонымен, десте желілік құрылғыны қабылдаудың сақиналық буферіне түскен кезде аппараттық үзіліс жасалады, оны SoftIRQ құрылғысының драйвері ksoftirqd процессімен өңделетін NET_RX_SOFTIRQ сияқты оқиға жасайды, ол өз кезегінде net_rx_action() функциясын шақырады, онда драйвер әдісінің poll() қоңырауы болады. Trace-cmd көмегімен біз қоңыраулар тізбегін анықтау үшін аздап тазартылған келесі нәтижені аламыз (E қосымшасы).

Бұратталғаннан кейін _raw_spin_lock_irqsave() десте пайдаланушы кеңістігі розеткасының буферіне көшіріледі, содан кейін _raw_spin_unlock_irqrestore() құлпын ашады, содан кейін sock_def_readable() құлыпты қайта шақырады _raw_spin_lock_irqsave() және розеткадағы дестетерді күту процесі ep_poll_callback() көмегімен дестетерді тиісті файлдардан алады Розетка дескрипторы және мұның бәрі бір CPU-да болады. Perf тұжырымы (3.9-суретқараңыз) әрбір қоңыраудан туындаған overhead ("үстеме шығындар") көрсетеді, көріп отырғаныңыздай, _raw_spin_unlock_ irqrestore() құлпы ең көп орын алады.

```
Samples: 111K of event 'cycles:pp', Event count (approx.): 60168447233
Overhead  Shared Object                     Symbol
   7.00%  [kernel]                          [k] _raw_spin_lock_irqsave
   3.58%  [kernel]                          [k] copy_user_generic_string
   3.49%  [kernel]                          [k] __udp4_lib_lookup
   3.33%  [kernel]                          [k] fib_table_lookup
   3.03%  [kernel]                          [k] __skb_recv_datagram
   2.70%  [kernel]                          [k] native_write_msr_safe
   2.54%  [kernel]                          [k] _raw_spin_lock
   2.53%  [kernel]                          [k] udp_recvmsg
   2.46%  [kernel]                          [k] sock_queue_rcv_skb
   2.36%  [kernel]                          [k] sk_run_filter
   2.28%  [kernel]                          [k] ixgbe_clean_rx_irq
   2.24%  [kernel]                          [k] _raw_spin_unlock_irqrestore
   2.24%  [kernel]                          [k] _raw_spin_lock_bh
   2.21%  [kernel]                          [k] __memcpy
   2.07%  [kernel]                          [k] __netif_receive_skb_core
   1.89%  [kernel]                          [k] __schedule
   1.72%  [kernel]                          [k] __udp4_lib_rcv
```

3.9 сурет – UDP дестетерін алған кезде perf шығару бөлігі

3.3 NETMAP фреймворкын тестілеу

NETMAP фреймворкын сынауды бастау үшін оны жинау керек. Әрі қарайғы әрекеттер екі серверде жасалады.

Біз NETMAP репозиторийін клондаймыз, ixgbe драйверін пайдаланып құрастыруды конфигурациялаймыз, құрастырамыз және жинаймыз:

git clone https://github.com/luigirizzo/netmap.git
cd netmap/
./configure --drivers=ixgbe
make

Ескі ixgbe драйвер модулін ядродан түсіріп, netmap.ko және ixgbe.ko модульдерін жүктеңіз:

rmmod ixgbe
insmod ./netmap.ko
insmod ./ixgbe/ixgbe.ko

Желілік интерфейсті қосыңыз:

ip link set enp4s0f0 up

NETMAP TS, UFA, RX/TX checksum offloading және т. б. қолдамайды, сондықтан оларды өшіру керек:

ethtool -K enp4s0f0 tx off rx off gso off tso off gro off lro off

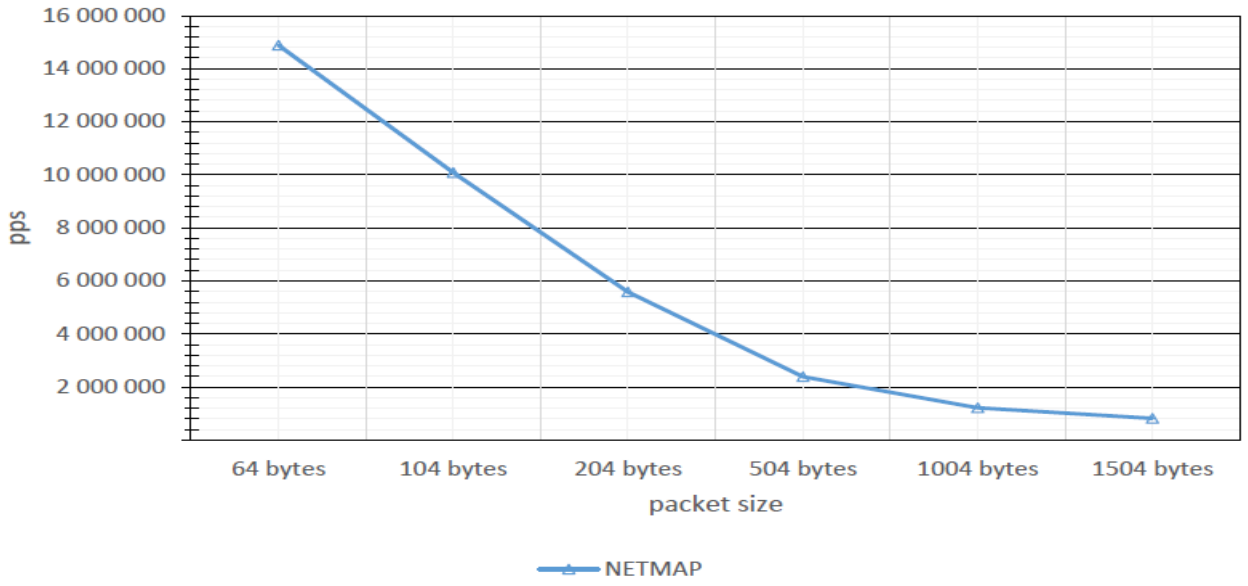
Дестетерді құру және қабылдау үшін NETMAP репозиторийіне кіретін pkt-gen утилитасы қолданылады. Бірінші серверде 64, 104, 204, 504, 1004, 1504 байт максималды дестетік жылдамдықта Ethernet рамалары жасалады. Екінші серверде сол қызметтік бағдарламаны пайдаланып қабылданған дестетердің саны есептеледі. Өлшеу нәтижелері 3.7-кестеге енгізілген. Орташа мәндер үшін тиісті графиктер салынған (3.10 және 3.11-суреттерді қараңыз).

3.7 кесте – NETMAP көмегімен дестетерді басып алу өнімділігі

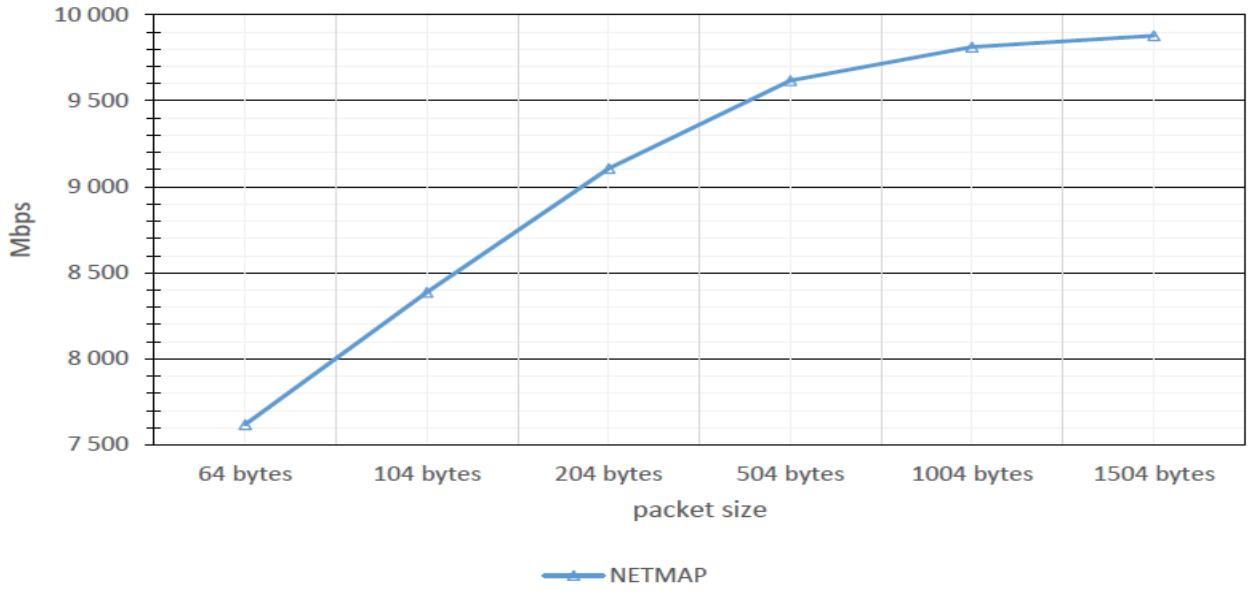
№	64 bytes	104 bytes	204 bytes	504 bytes	1004 bytes	1504 bytes
1	14 878 351	10 080 115	5 580 554	2 385 115	1 221 988	821 033
2	14 878 851	10 080 918	5 580 314	2 385 525	1 221 474	821 051
3	14 878 354	10 080 354	5 580 358	2 385 654	1 221 774	821 016
4	14 878 153	10 080 165	5 580 918	2 385 382	1 221 817	821 029
5	14 878 154	10 080 355	5 580 133	2 385 181	1 221 835	821 045
6	14 878 154	10 080 595	5 580 747	2 385 119	1 221 657	821 020

3.7 кестенің жалғасы

№	64 bytes	104 bytes	204 bytes	504 bytes	1004 bytes	1504 bytes
7	14 878 385	10 080 566	5 580 132	2 385 354	1 221 811	821 048
8	14 878 985	10 080 465	5 580 984	2 385 546	1 221 824	821 020
9	14 878 544	10 080 181	5 580 476	2 385 354	1 221 935	821 046
10	14 879 352	10 080 313	5 580 359	2 385 185	1 221 254	821 021
Avg.pps	14 878 568	10 080 403	5 580498	2 385 342	1 221 737	821 033
Avg.Mbps	7 617 827	8 386 895	9 107 373	9 617 699	9 812 992	9 878 669

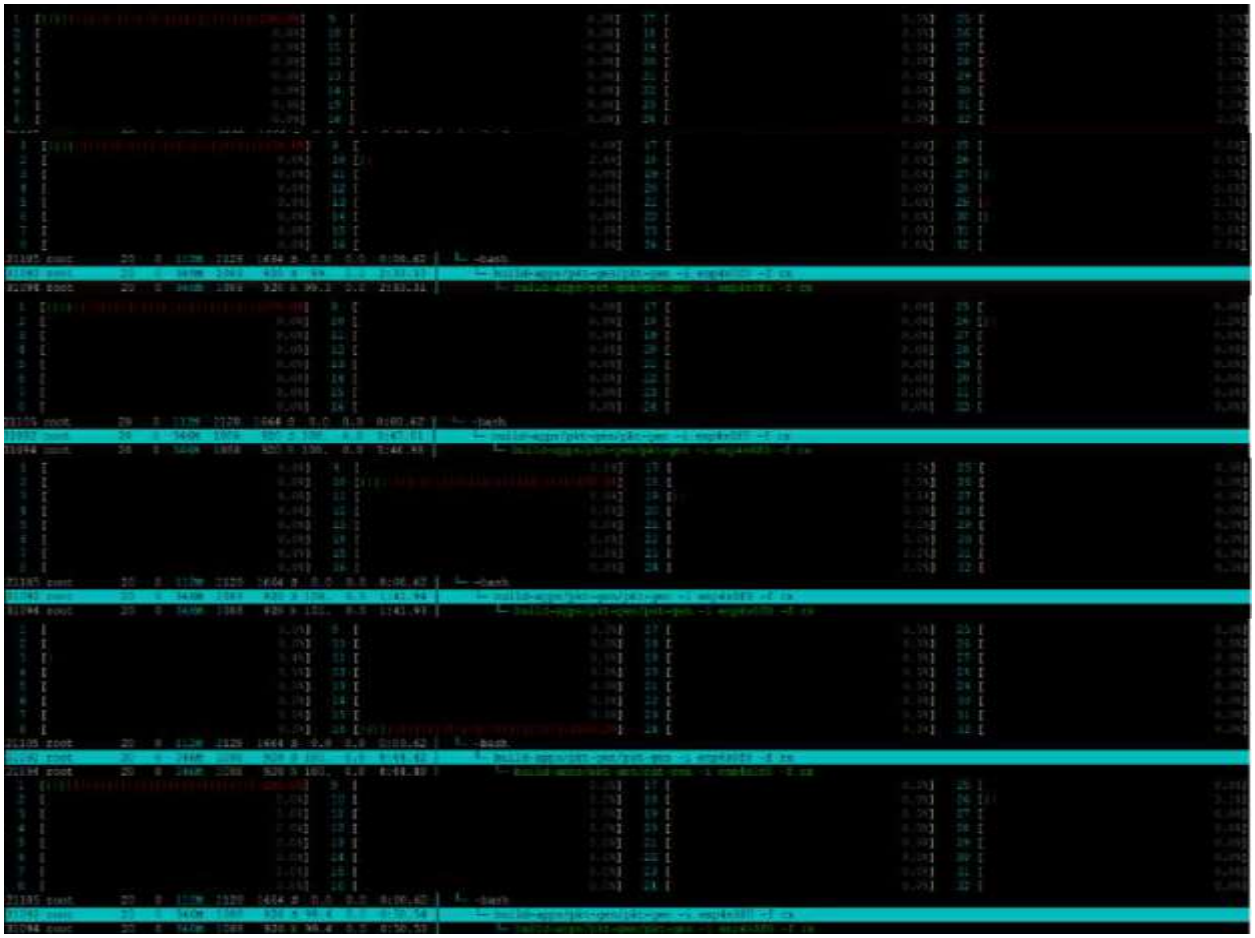


3.10 сурет – PF_RING пайдалану кезінде дестетік басып алу өнімділігінің орташа графигі

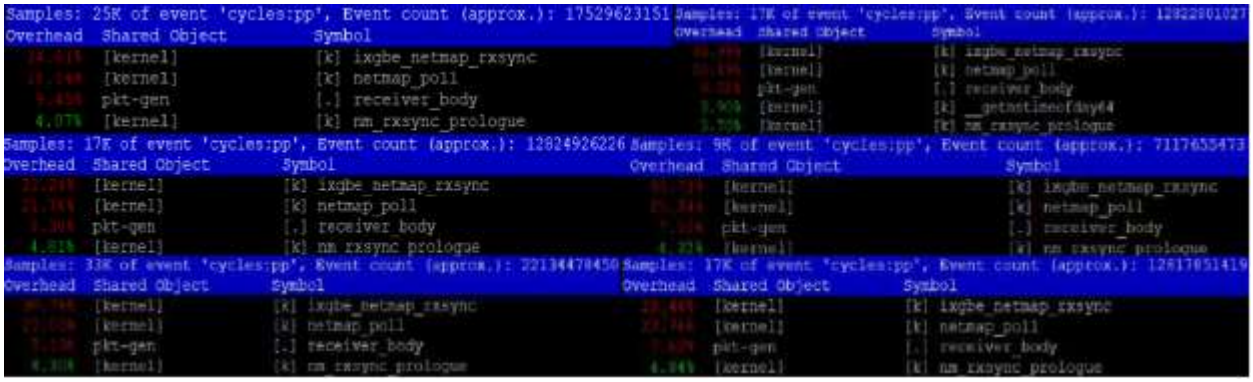


3.11 сурет – NETMAP пайдалану кезінде өңделетін трафик көлемінің орташа графигі

Luigi Rizzo суреттегендей, NETMAP көмегімен 1 GHz + бір ядросында (3.12-суретті қараңыз) 14.8 Mpps басып алуға оңай қол жеткізуге болады. Perf қорытындысынан (3.13-суретті қараңыз) Linux желілік стекіне әсер етпейтінін көруге болады.



3.12 сурет – NETMAP көмегімен дестетерді басып алу кезінде htop шығару бөлігі (жоғарыдан төменге қарай 64-1504 байт)



3.13 сурет – NETMAP көмегімен дестетерді басып алу кезінде perf шығару бөлігі (жоғарыдан төмен қарай солдан оңға қарай 64-1504 байт)

3.4 PF_RING кітапханасын тестілеу

Жалпыға қол жетімді PF_RING-тің әдеттегі нұсқасы сыналады.

Біз PF_RING репозиторийін клондаймыз, құрастырамыз және жинаймыз:

git clone https://github.com/ntop/PF_RING.git
cd PF_RING/kernel/
make

Модульді ядроға жүктеңіз:

insmod pf_ring.ko min_num_slots=4096 enable_tx_capture=0 quick_mode=1

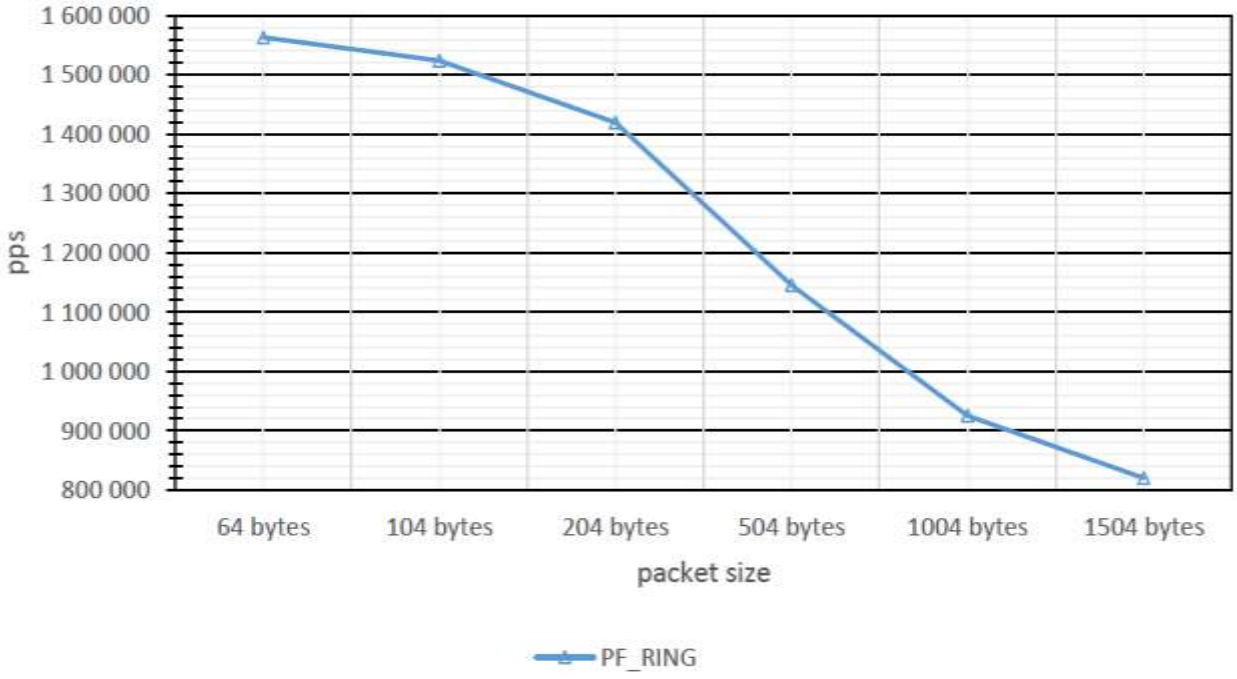
PF_RING пайдаланатын және қабылданған дестетердің санын есептейтін сол репозиторийде pfcount утилитасын дайындаймыз:

cd ../userland
./configure
make

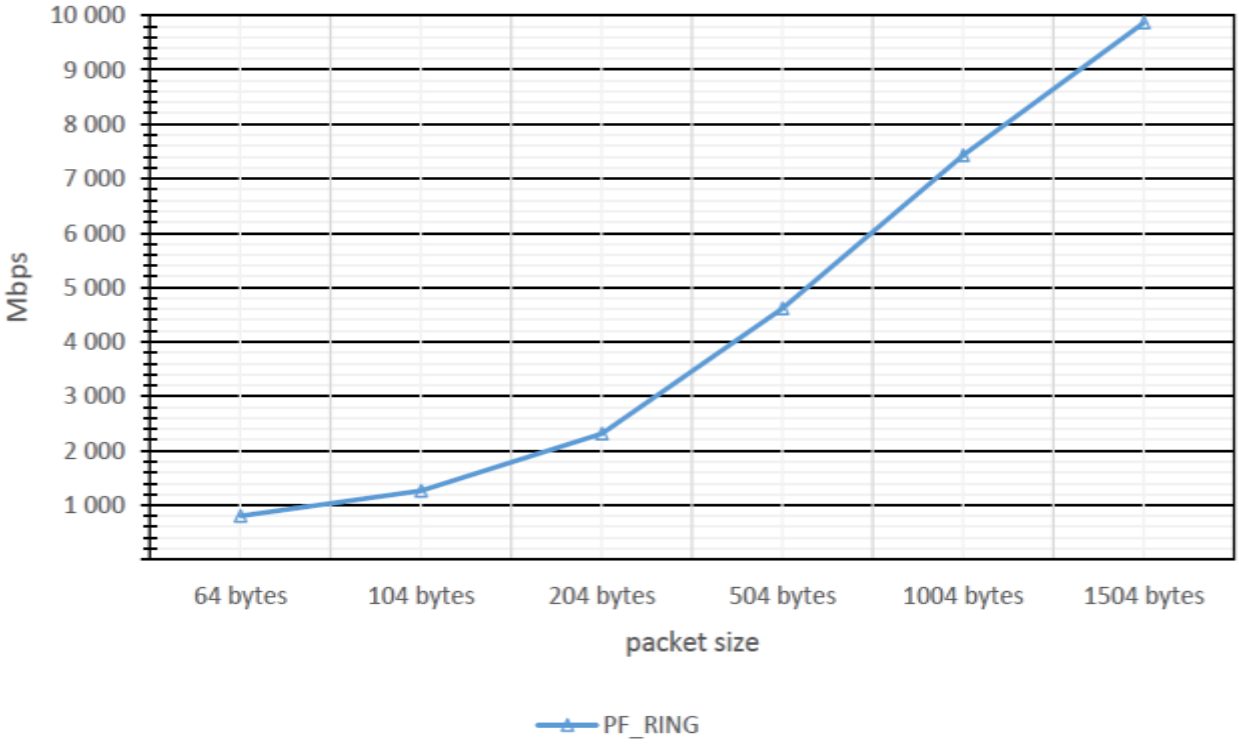
Бірінші серверде 64, 104, 204, 504, 1004, 1504 байт максималды дестетік жылдамдықта Ethernet рамалары жасалады. Екінші серверде pf count утилитасы қабылданған дестетердің санын есептейді. Өлшеу нәтижелері 3.8-кестеге енгізілген. Орташа мәндер үшін тиісті графиктер салынған (3.14 және 3.15-суреттерді қараңыз).

3.8 кесте – PF_RING көмегімен дестетерді басып алу өнімділігі

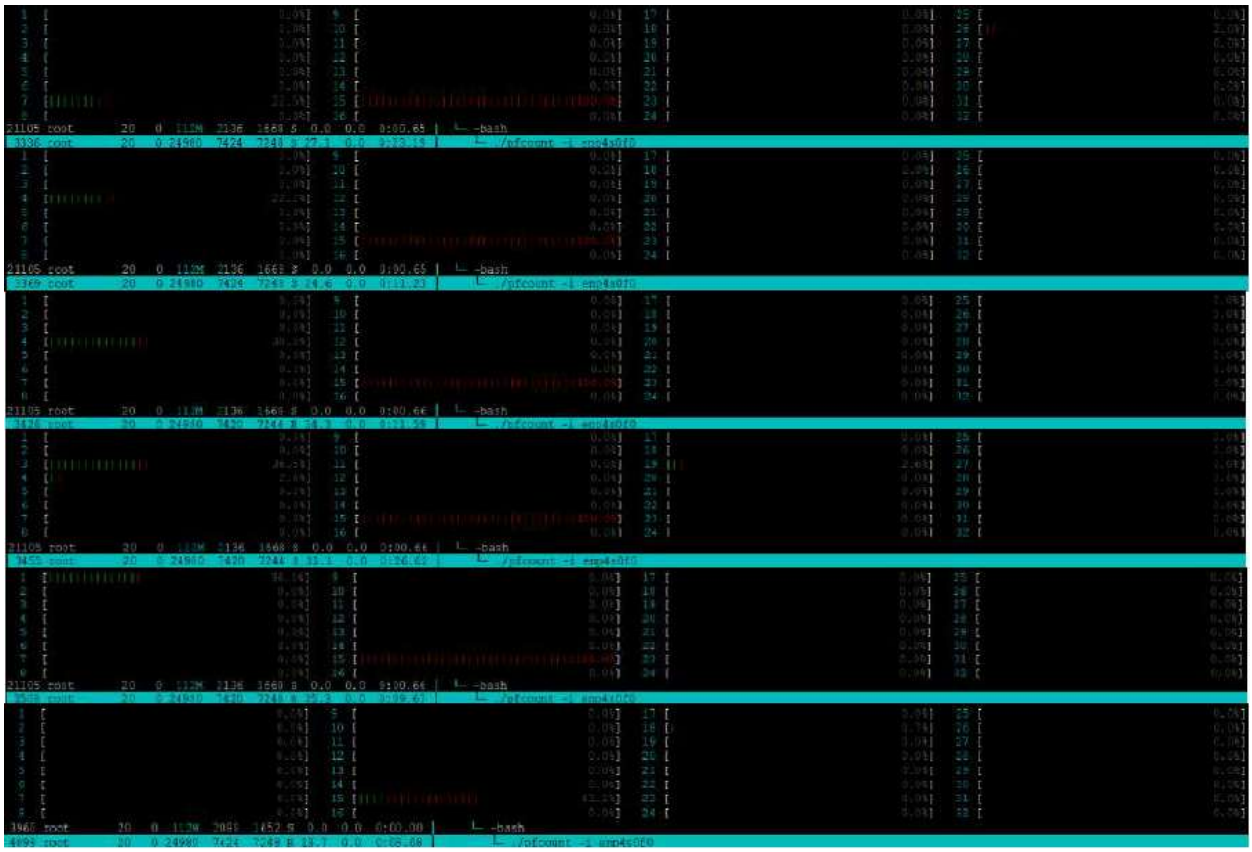
№	64 bytes	104 bytes	204 bytes	504 bytes	1004 bytes	1504 bytes
1	1 510 266	1 656 672	1 418 787	1 143 135	925 022	820 257
2	1 507 833	1 468 986	1 418 486	1 141 794	925 285	820 172
3	1 511 582	1 674 043	1 417 915	1 141 048	924 919	820 261
4	1 515 713	1 437 963	1 421 164	1 140 620	924 965	820 261
5	1 511 283	1 472 040	1 422 945	1 144 112	925 079	820 261
6	1 511 502	1 443 459	1 423 186	1 138 038	923 798	820 422
7	1 770 312	1 458 355	1 415 751	1 143 413	925 614	820 395
8	1 511 019	1 714 651	1 417 751	1 140 318	925 368	820 031
9	1 510 261	1 465 002	1 420 812	1 143 590	925 159	820 201
10	1 772 723	1 452 265	1 418 544	1 173 841	925 030	820 143
Avg.pps	1 563 249	1 524 344	1 419 534	1 144 991	925 024	820 215
Avg.Mbps	800 384	1 268 254	2 316 679	4 616 604	7 429 793	9 868 827



3.14 сурет – PF_RING пайдалану кезінде дестетік басып алу өнімділігінің орташа графигі

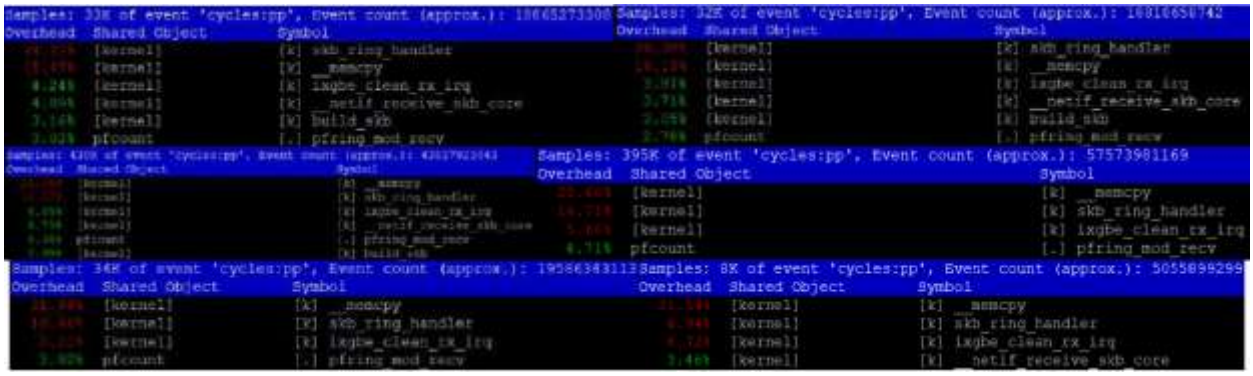


3.15 сурет – PF_RING пайдалану кезінде өңделетін трафик көлемінің орташа мәндерінің графигі



3.16 сурет – Дестетерді PF_RING көмегімен басып алу кезінде htop шығару бөлігі (жоғарыдан төменге қарай 64-1504 байт)

Күтілгендей, бір ядрода PF_RING (ZC емес) әлсіз нәтиже берді, бірақ стандартты Linux құралдарына қарағанда шамамен 0.5 m 64 байт дестетерге көп. Htop және perf статистикасы сәйкесінше 3.16 және 3.17 суреттерінде келтірілген.



3.17 сурет – Дестетерді PF_RING көмегімен басып алу кезінде perf шығару бөлігі (жоғарыдан төмен қарай солдан оңға қарай 64-1504 байт)

3.5 DPDK фреймворкын тестілеу

DPDK көздерін жүктеп, оларды жинауға дайындаңыз:

wget https://fast.dpdk.org/rel/dpdk-17.11.1.tar.xz
tar -xf dpdk-17.11.1.tar.xz
cd dpdk-stable-17.11.1/
make config T=x86_64-native-linuxapp-gcc

x86_64-nativelinuxapp-gcc/.config файлына орнату арқылы HPET қолдауын қосыңыз. CONFIG_RTE_LIBEAL_USE_HPET=y параметрін жүктеу кезінде BIOS-қа алдын-ала қосыңыз.

.Біз құрастырамыз, жинаймыз және орнатамыз:

make install T=x86_64-native-linuxapp-gcc

Біз ағымдағы hugepages-ті бөлшектейміз, каталог жасаймыз және оған hugepages орнатамыз:

umount `awk '/hugetlbfs/ {print $2}' /proc/mounts`
mkdir -p /mnt/huge
mount -t hugetlbfs nodev /mnt/huge

uio_pci_generic модулін жүктеңіз:

modprobe uio_pci_generic

Жүйеде желілік порттардың күйін тексеріңіз:

./usertools/dpdk-devbind.py –status
> Network devices using kernel driver
> ==========
> 0000:04:00.0 '82599ES 10-Gigabit SFI/SFP+ Network Connection 10fb' if=enp4s0f0 drv=ixgbe unused=
> 0000:04:00.1 '82599ES 10-Gigabit SFI/SFP+ Network Connection 10fb' if=enp4s0f1 drv=ixgbe unused=
*> 0000:07:00.0 'I350 Gigabit Network Connection 1521' if=eth0 drv=igb unused= *Active**
> 0000:07:00.1 'I350 Gigabit Network Connection 1521' if=eno1 drv=igb unused=

Domain 0, Bus 0, Device 4, Function 0 (0000:04:00.0) uio_pci_generic драйвері бар PCI құрылғысына қосылыңыз:

./usertools/dpdk-devbind.py --bind=uio_pci_generic 04:00.0

Біз DPDK құрамына кіретін test-pmd утилитасын пайдаланып дестетер санын есептейміз. Біз test-pmd құрастырамыз және жинаймыз:

export RTE_SDK=/root/dpdk-stable-17.11.1/
export RTE_TARGET=x86_64-native-linuxapp-gcc
cd app/test-pmd/
make

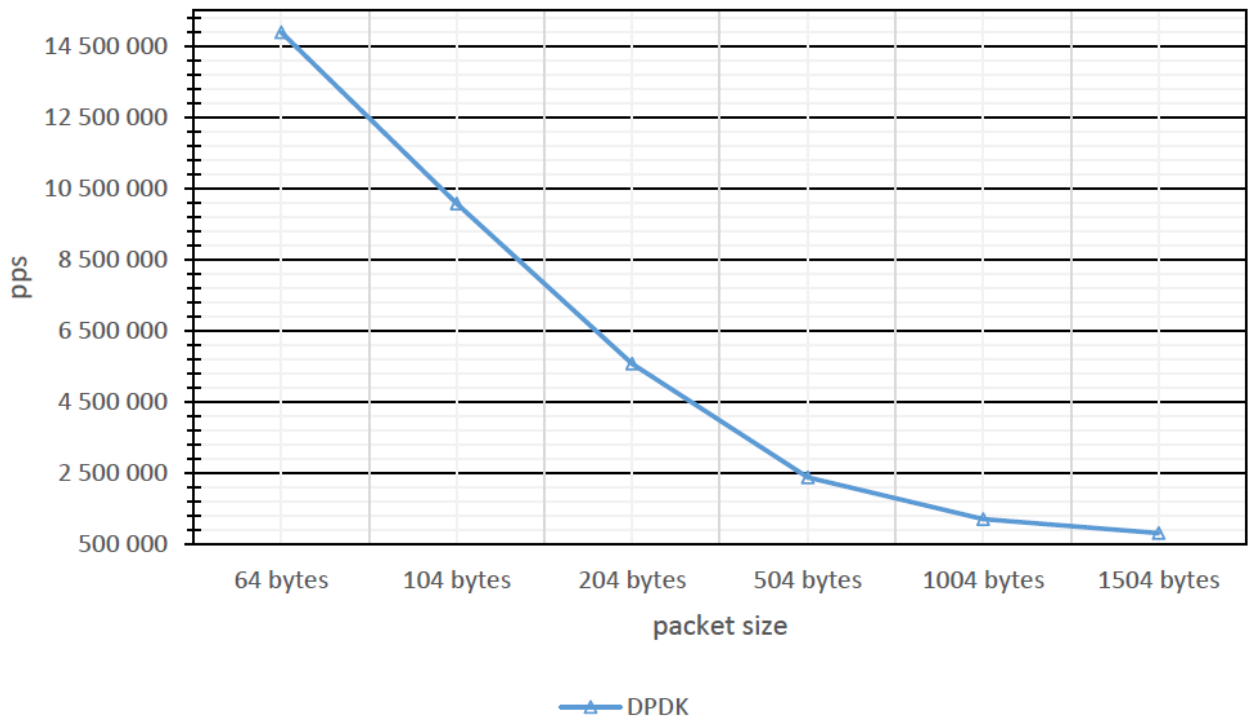
Тест-pmd іске қосамыз:

shell> ./testpmd –l 6 –n 4 -- -i --huge-dir /mnt/huge
testpmd> start

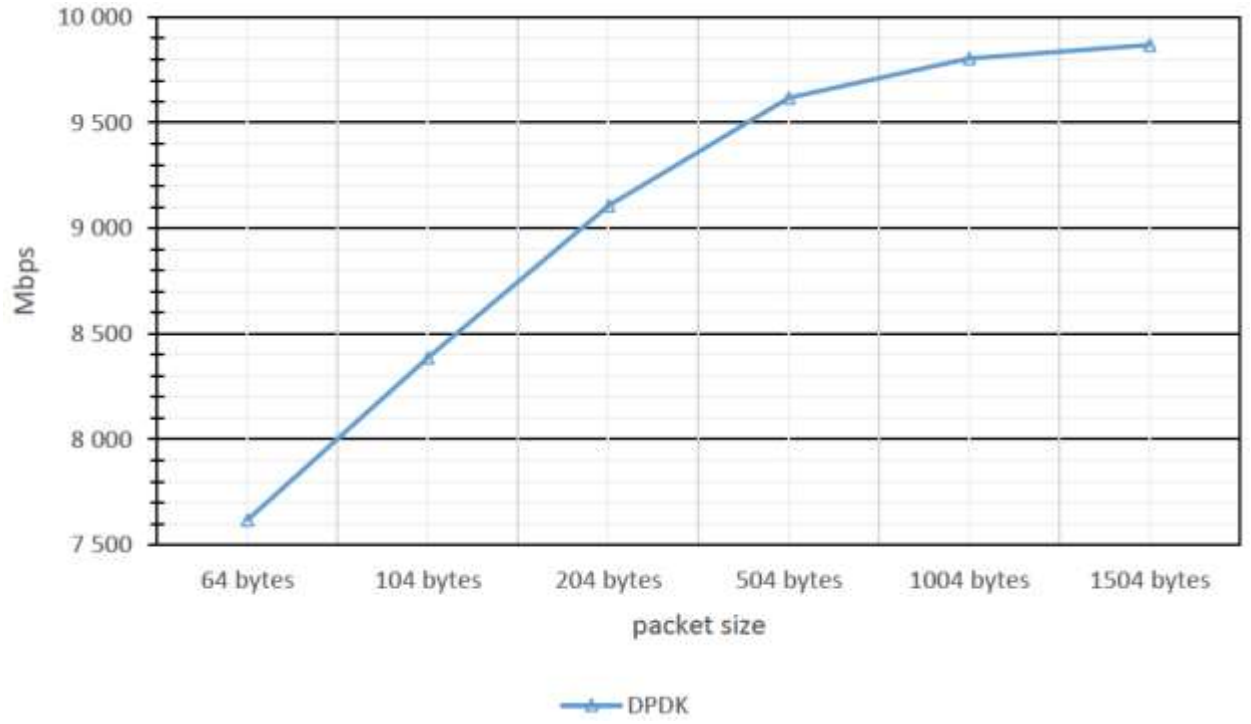
Бірінші серверде 64, 104, 204, 504, 1004, 1504 байт максималды дестетік жылдамдықта Ethernet рамалары жасалады. Екінші серверде test-pmd утилитасы қабылданған дестетердің санын есептейді. Өлшеу нәтижелері 3.9-кестеге енгізілген. Орташа мәндер үшін тиісті графиктер салынған (3.18 және 3.19-суреттерді қараңыз).

3.9 кесте – DDK көмегімен дестетерді басып алу өнімділігі

№	64 bytes	104 bytes	204 bytes	504 bytes	1004 bytes	1504 bytes
1	14 879 859	10 080 032	5 579 996	2 385 349	1 220 624	820 162
2	14 880 336	10 079 992	5 579 993	2 385 348	1 220 623	820 158
3	14 880 317	10 079 992	5 580 000	2 385 323	1 220 628	820 158
4	14 879 962	10 080 031	5 580 004	2 385 323	1 220 628	820 158
5	14 879 986	10 079 957	5 579 999	2 385 344	1 220 616	820 154
6	14 871 062	10 081 124	5 579 989	2 385 281	1 220 629	820 160
7	14 881 854	10 079 881	5 579 986	2 385 349	1 220 623	820 155
8	14 886 841	10 079 899	5 579 996	2 385 411	1 220 625	820 161
9	14 879 174	10 081 185	5 580 001	2 385 366	1 220 619	820 157
10	14 879 355	10 081 447	5 579 995	2 385 315	1 220 618	820 159
Avg.pps	14 879 875	10 080 354	5 579 996	2 385 349	1 220 623	820 158
Avg.Mbps	7 618 496	8 386 855	9 106 553	9 617 727	9 804 044	9 868 141

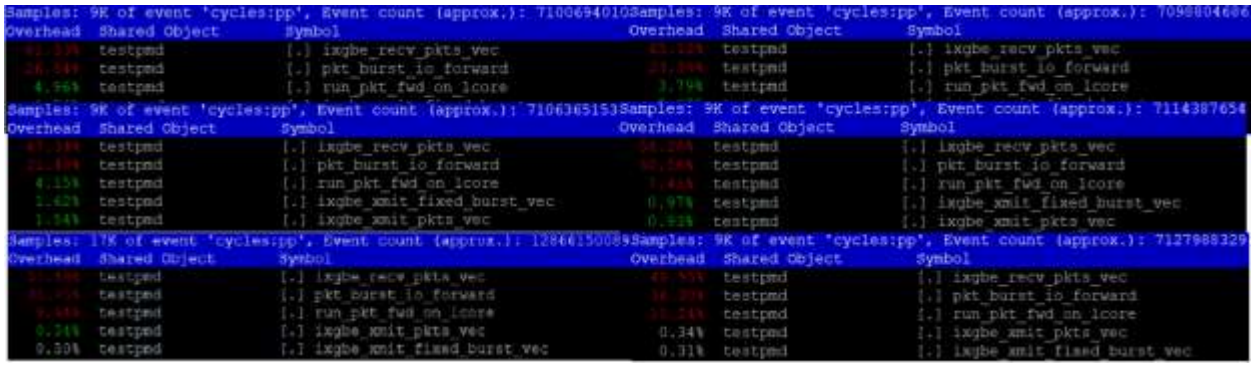


3.18 сурет – DPDK пайдалану кезінде дестетік басып алу өнімділігінің орташа графигі

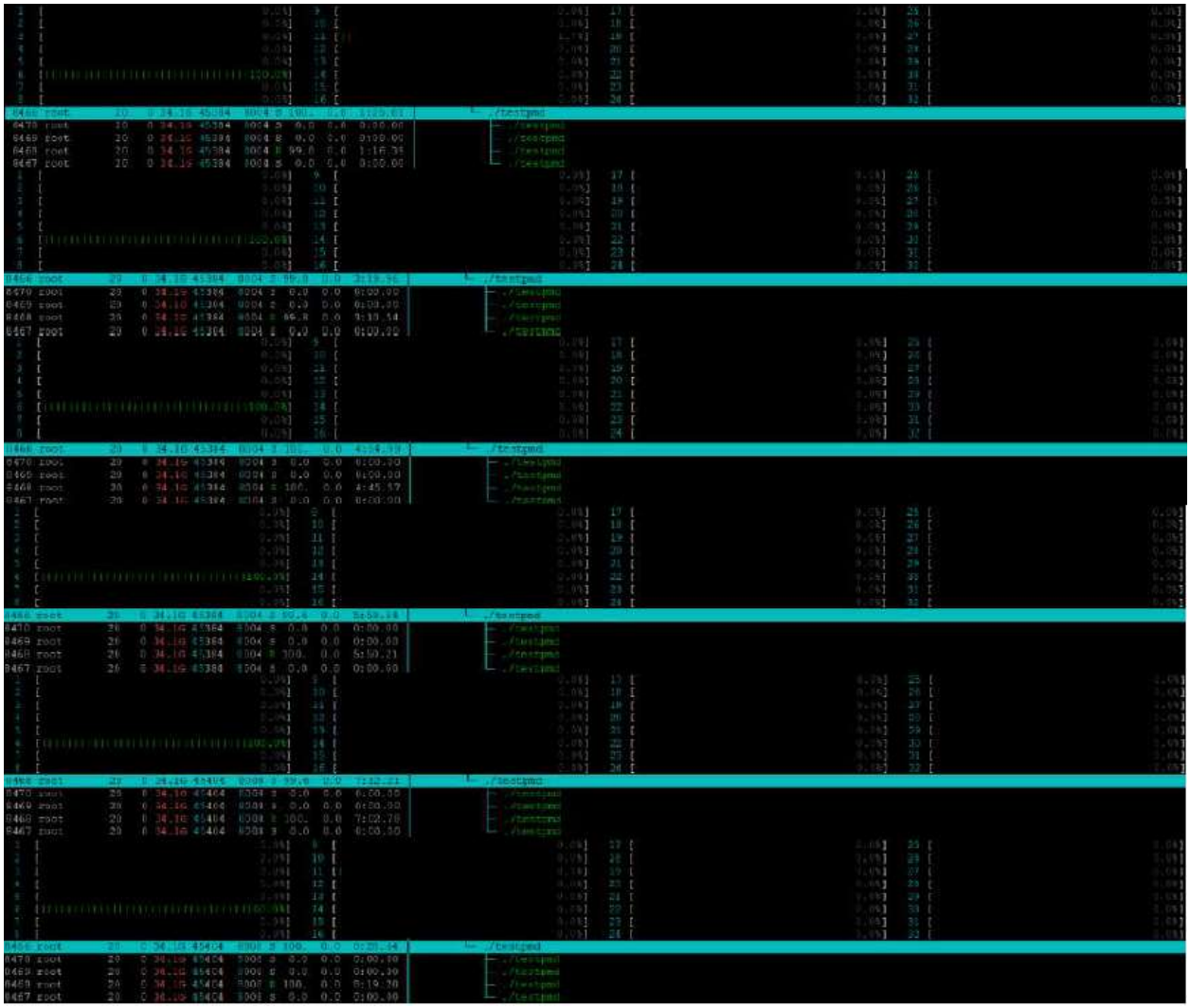


3.19 сурет – DPDK пайдалану кезінде өңделетін трафик көлемінің орташа графигі

Нәтижелер көрсеткендей, бір ядрода (3.21-суретті қараңыз) DPDK фреймворкын пайдаланып, Linux желілік стекін айналып өтіп, 14.8 Mpps басып алуға қол жеткізуге болады (3.20-суретті қараңыз).



3.20 сурет – Дестетерді DPDK көмегімен басып алу кезінде perf шығару бөлігі (жоғарыдан төмен қарай солдан оңға қарай 64-1504 байт)



3.21 сурет – DPDK көмегімен дестетерді басып алу кезінде htop шығару бөлігі (жоғарыдан төменге қарай 64-1504 байт)

3.6 nDPI кітапханасын тестілеу

nDPI репозиторийін клондау, құрастыру және жинау:

git clone https://github.com/ntop/nDPI.git
cd nDPI/
./autogen.sh
make

nDPI репозиторийіне ndpiReader утилитасы кіреді, ол nDPI кітапханасын пайдаланады және желілік құрылғыдан немесе pcap файлынан дестетерді талдай алады. Осы қызметтік бағдарламаның көмегімен біз pcap файлдарын тексереміз (Microsoft Message Analyzer және Wireshark көмегімен дайындалған және осы жерден алынған), трафикті құратын протоколдар мен қосымшалар туралы алдын-ала белгілі ақпарат, сонымен қатар flow тестілерін жүргіземіз. Трафик талданған бағдарламалық жасақтама: qbittorrent, Skype, Steam.

Протколдар: SSH, SSL, SMTP.

Тиісті pcap файлдары үшін ndpiReader қорытынды:

- SSH.pcap (20 pkts, 4120 bytes, 1 flow):

...
Traffic statistics:
Ethernet bytes: 4600 (includes ethernet CRC/IFC/trailer)
Discarded bytes: 0
IP packets: 20 of 20 packets total
IP bytes: 4120 (avg pkt size 206 bytes)
Unique flows: 1
TCP Packets: 20
...
Detected protocols:
SSH packets: 20 bytes: 4120 flows: 1

- SSL.pcap (17 pkts, 3231 bytes, 1 flow):

...
Traffic statistics:
Ethernet bytes: 3639 (includes ethernet CRC/IFC/trailer)
Discarded bytes: 0
IP packets: 17 of 17 packets total
IP bytes: 3231 (avg pkt size 190 bytes)
Unique flows: 1
TCP Packets: 17
...
Detected protocols:
SSL packets: 17 bytes: 3231 flows: 1

- SMTP.pcap (20 pkts, 4514 bytes, 1 flow):

...
Traffic statistics:
Ethernet bytes: 4994 (includes ethernet CRC/IFC/trailer)
Discarded bytes: 0
IP packets: 20 of 20 packets total
IP bytes: 4514 (avg pkt size 225 bytes)
Unique flows: 1
TCP Packets: 20
...
Detected protocols:
Unknown packets: 20 bytes: 4514 flows: 1

- qBittorrent.pcap (17836 pkts, 2896888 bytes):

...
Traffic statistics:
Ethernet bytes: 3324784 (includes ethernet CRC/IFC/trailer)
Discarded bytes: 120
IP packets: 17834 of 17836 packets total
IP bytes: 2896768 (avg pkt size 162 bytes)
Unique flows: 1625
TCP Packets: 5616
UDP Packets: 12212
...
Detected protocols:
Unknown packets: 4840 bytes: 501270 flows: 665
DNS packets: 33 bytes: 3327 flows: 11
HTTP packets: 1161 bytes: 188276 flows: 90
NetBIOS packets: 6 bytes: 552 flows: 1
BitTorrent packets: 9511 bytes: 1554137 flows: 762
IGMP packets: 3 bytes: 162 flows: 1
SSL packets: 1363 bytes: 348859 flows: 57
Google packets: 330 bytes: 124436 flows: 7
UPnP packets: 80 bytes: 11360 flows: 6
Tor packets: 122 bytes: 75163 flows: 3
Amazon packets: 88 bytes: 10786 flows: 10
QUIC packets: 4 bytes: 248 flows: 1
Cloudflare packets: 293 bytes: 78192 flows: 11

Толық нәтиже сілтеме бойынша қол жетімді.
- Skype.pcap (2434 pkts, 1099982 bytes):

...
Traffic statistics:

Ethernet bytes: 1158234 (includes ethernet CRC/IFC/trailer)
Discarded bytes: 116
IP packets: 2432 of 152 packets total
IP bytes: 1099866 (avg pkt size 451 bytes)
Unique flows: 77
TCP Packets: 1061
UDP Packets: 1189
...
Detected protocols:
Unknown packets: 136 bytes: 57072 flows: 7
HTTP packets: 16 bytes: 1232 flows: 1
SSDP packets: 12 bytes: 2094 flows: 2
STUN packets: 136 bytes: 68524 flows: 5
ICMP packets: 182 bytes: 13516 flows: 6
SSL packets: 212 bytes: 106667 flows: 4
Skype packets: 1606 bytes: 710953 flows: 51
Microsoft packets: 132 bytes: 139808 flows: 1

Толық нәтиже сілтеме бойынша қол жетімді.

- Steam.pcap (51005 pkts, 54684173 bytes):

...
Traffic statistics:
Ethernet bytes: 55898651 (includes ethernet CRC/IFC/trailer)
Discarded bytes: 6762
IP packets: 50885 of 51005 packets total
IP bytes: 54677411 (avg pkt size 1072 bytes)
Unique flows: 87
TCP Packets: 50276
UDP Packets: 368
...
Detected protocols:
Unknown packets: 63 bytes: 8973 flows: 3
DNS packets: 48 bytes: 6556 flows: 6
HTTP packets: 332 bytes: 95812 flows: 7
MDNS packets: 72 bytes: 8136 flows: 2
NTP packets: 6 bytes: 540 flows: 1
NetBIOS packets: 18 bytes: 1656 flows: 1
SSDP packets: 192 bytes: 63792 flows: 4
DHCP packets: 10 bytes: 3420 flows: 1
Steam packets: 2088 bytes: 1527420 flows: 32
SSL packets: 47786 bytes: 52901970 flows: 15
SSH packets: 6 bytes: 480 flows: 1

Google packets: 208 bytes: 54672 flows: 4
LLMNR packets: 56 bytes: 3984 flows: 10

Толық нәтиже сілтеме бойынша қол жетімді.
- smallFlows.pcap (14261 pkts, 9216531 bytes, 1209 flows):

...
Traffic statistics:
Ethernet bytes: 9557445 (includes ethernet CRC/IFC/trailer)
Discarded bytes: 918
IP packets: 14243 of 14261 packets total
IP bytes: 9215613 (avg pkt size 646 bytes)
Unique flows: 635
TCP Packets: 13708
UDP Packets: 501
...
Detected protocols:
Unknown packets: 154 bytes: 19895 flows: 56
DNS packets: 40 bytes: 8900 flows: 10
HTTP packets: 1984 bytes: 1124491 flows: 133
NetBIOS packets: 43 bytes: 3954 flows: 4
SSDP packets: 42 bytes: 7229 flows: 6
SNMP packets: 16 bytes: 1952 flows: 1
SMB packets: 16 bytes: 1968 flows: 1
DHCP packets: 3 bytes: 1026 flows: 3
Flash packets: 292 bytes: 287997 flows: 2
SSL_No_Cert packets: 355 bytes: 45337 flows: 12
MSN packets: 6106 bytes: 4562906 flows: 153
ICMP packets: 22 bytes: 4190 flows: 2
SSL packets: 2542 bytes: 2039243 flows: 28
Facebook packets: 119 bytes: 54792 flows: 8
Dropbox packets: 16 bytes: 2592 flows: 2
Skype packets: 1146 bytes: 575773 flows: 134
Google packets: 1104 bytes: 394392 flows: 63
LLMNR packets: 4 bytes: 256 flows: 2
Microsoft packets: 112 bytes: 56208 flows: 3
Office365 packets: 33 bytes: 3287 flows: 3
OpenDNS packets: 94 bytes: 19225 flows: 9

Толық нәтиже сілтеме бойынша қол жетімді.
- bigFlows.pcap (791615 pkts, 355417784 bytes, 40686 flows):

...
Traffic statistics:

Ethernet bytes: 374416544 (includes ethernet CRC/IFC/trailer)
Discarded bytes: 0
IP packets: 791615 of 791615 packets total
IP bytes: 355417784 (avg pkt size 448 bytes)
Unique flows: 28062
TCP Packets: 633894
UDP Packets: 153135
...
Detected protocols:
Unknown packets: 178386 bytes: 51159726 flows: 1545
FTP_CONTROL packets: 41 bytes: 2585 flows: 5
POP3 packets: 94 bytes: 66630 flows: 1
DNS packets: 3508 bytes: 362132 flows: 1448
HTTP packets: 217117 bytes: 66144474 flows: 15752
MDNS packets: 26 bytes: 9330 flows: 10
NTP packets: 4 bytes: 360 flows: 2
NetBIOS packets: 692 bytes: 78146 flows: 96
SSDP packets: 623 bytes: 113586 flows: 66
SNMP packets: 3418 bytes: 503638 flows: 1323
SMB packets: 30 bytes: 6425 flows: 2
Syslog packets: 599 bytes: 207546 flows: 1
DHCP packets: 62 bytes: 21993 flows: 19
EDonkey packets: 111 bytes: 43359 flows: 1
BitTorrent packets: 10 bytes: 1022 flows: 1
Flash packets: 807 bytes: 685991 flows: 10
QQ packets: 4 bytes: 392 flows: 1
HTTP_Download packets: 395 bytes: 383280 flows: 9
SSL_No_Cert packets: 6004 bytes: 4535852 flows: 60
Unencrypted_Jabber packets: 321 bytes: 211287 flows: 2
MSN packets: 191 bytes: 87114 flows: 17
Oscar packets: 43 bytes: 2962 flows: 2
Yahoo packets: 4081 bytes: 2445929 flows: 235
GooglePlus packets: 1440 bytes: 856224 flows: 22
ICMP packets: 3619 bytes: 312455 flows: 532
IGMP packets: 286 bytes: 18148 flows: 100
SSL packets: 205599 bytes: 132136495 flows: 2892
SSH packets: 4306 bytes: 541862 flows: 24
SIP packets: 40 bytes: 28718 flows: 1
DHCPV6 packets: 361 bytes: 57327 flows: 62
Facebook packets: 4720 bytes: 1999215 flows: 236
Twitter packets: 1576 bytes: 574405 flows: 155
Dropbox packets: 473 bytes: 92121 flows: 31
Gmail packets: 1758 bytes: 959432 flows: 27
GoogleMaps packets: 520 bytes: 419797 flows: 10

YouTube packets: 24104 bytes: 26408966 flows: 115
Skype packets: 2071 bytes: 283926 flows: 165
Google packets: 36543 bytes: 19041044 flows: 1279
DCE_RPC packets: 34827 bytes: 20877856 flows: 115
sFlow packets: 1113 bytes: 1493278 flows: 1
HTTP_Proxy packets: 166 bytes: 57475 flows: 2
Citrix packets: 17445 bytes: 2005515 flows: 591
Apple packets: 10 bytes: 1757 flows: 1
AppleiCloud packets: 127 bytes: 26177 flows: 6
AppleiTunes packets: 2 bytes: 242 flows: 1
Radius packets: 95 bytes: 7552 flows: 63
WindowsUpdate packets: 7 bytes: 936 flows: 1
TeamViewer packets: 19 bytes: 1374 flows: 1
LLMNR packets: 58 bytes: 4292 flows: 29
Spotify packets: 34 bytes: 2924 flows: 6
H323 packets: 635 bytes: 47200 flows: 1
SOCKS packets: 4 bytes: 240 flows: 1
RTMP packets: 1201 bytes: 1297954 flows: 4
Amazon packets: 15786 bytes: 7256419 flows: 645
CNN packets: 135 bytes: 91623 flows: 9
Pandora packets: 3778 bytes: 3860971 flows: 9
Ookla packets: 16 bytes: 9292 flows: 1
Microsoft packets: 51 bytes: 11418 flows: 5
Office365 packets: 95 bytes: 12525 flows: 13
Cloudflare packets: 373 bytes: 188038 flows: 9
MS_OneDrive packets: 5 bytes: 308 flows: 1
MQTT packets: 40 bytes: 2605 flows: 5
RX packets: 2 bytes: 410 flows: 1
LinkedIn packets: 11040 bytes: 7045656 flows: 252
GoogleServices packets: 566 bytes: 309527 flows: 29
GoogleDocs packets: 2 bytes: 326 flows: 1

Толық нәтиже сілтеме бойынша қол жетімді.

nDPI кітапханасының көмегімен SSH, SSL және SMTP протоколдарының барлық дестетерін тануға болады. Таңдалған қолданбалар деректерді беру үшін әртүрлі протоколдарды пайдаланады (ерекше және стандартты). БЖ qBittorrent үшін дестетердің жалпы санынан 27.14%, Skype үшін – 5.59%, Steam үшін – 0.91% тану мүмкін болмады.

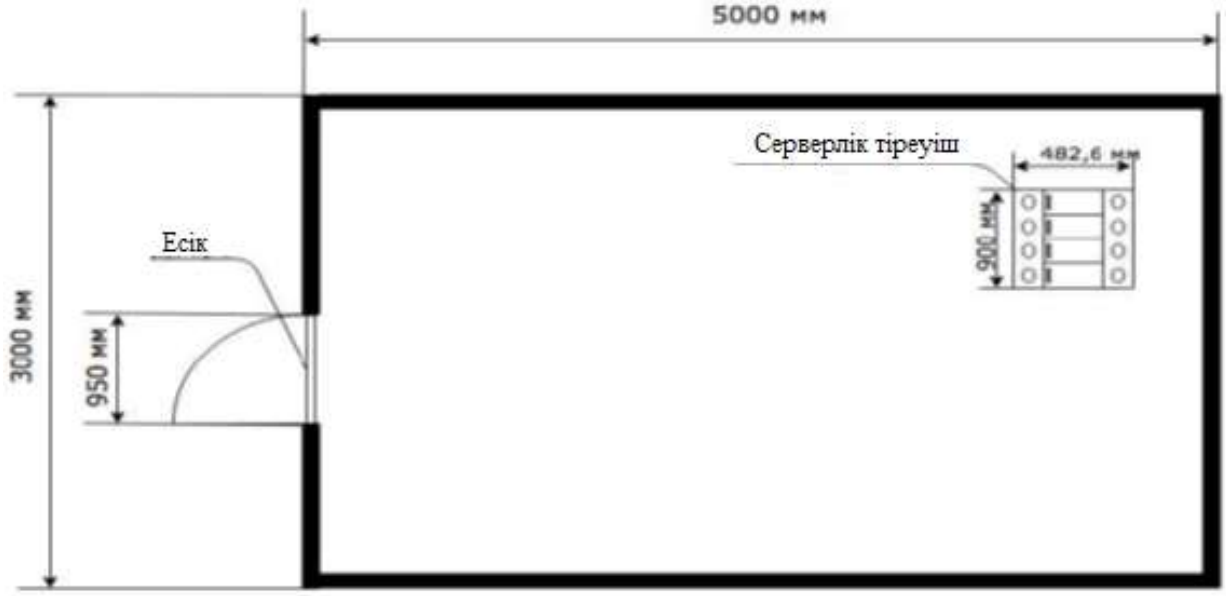
Small flow үшін тест дестетердің 1.08% және трафик ағындарының 52.52%, bigflow үшін дестетердің 22.53% және трафик ағындарының 68.97% танылмады.

4 Өмір тіршілік қауіпсіздігі

4.1 Еңбек жағдайларын талдау

Дипломдық жобаның мәні уақыт бірлігіне желілік дестетердің көп мөлшерін алудың қол жетімді технологияларын қарастыру және оларды кейінгі талдау болып табылады. Жалпы алғанда, оның көрінісінде қарастырылған технологиялардың жиынтығы сервер бөлмесінде орналасқан Бірлік өлшеміндегі серверге орнатылады және конфигурацияланады деп болжанады.

Серверлік бөлменің (4.1-суретті қараңыз) Алматы қаласында орналасқан және ауданы 15 м^2 және көлемі 45 м^3 болатын терезелері жоқ бөлмені білдіреді. Серверлік үй-жайдың қажетті жағдайларына жеткілікті жарықтандыру, электр қауіпсіздігі мен өрт қауіпсіздігін қамтамасыз ету, техникалық жабдықтың жұмысы үшін оңтайлы микроклимат пен тазалық жасау жатады.



4.1 сурет – Серверлік бөлменің орналасуы

ANSI/TIA/EIA-569-C "телекоммуникациялық жолдар мен коммерциялық ғимараттар қауымдастығының стандарттары" стандартына сүйене отырып, серверлік шкафтан телекоммуникациялық розеткаға телекоммуникациялық кабельдер салынған көлден жер асты трассаларын пайдалану арқылы жобалануы керек. Көлденең трассалар жүйесі кез келген телекоммуникациялық тарату ортасын пайдалануды ескерте отырып жобаланады. Жолдың көлемін анықтау кезінде кабельдердің мөлшері, саны, Болашақ кеңейтуге арналған қорда ескеріліп, иіру радиусына қойылатын талаптар ескеріледі.

Жер жолдары ұялы еденнің аясында орналасқан (4.2-сұретті қараңыз), оның элементтері еденнің бетон плитасын қолдайтын жақтау қызметінде

қызмет етеді, ал ұйымдар тарату қораптарында қолданылады. Коллекторлық арналардықұяшықтарға қатысты дұрыс бұрыстарда орнатылады және бетон құрастырылады.

Телекоммуникациялық шкафтар серверлерді, телекоммуникациялық жабдықты орнату және орнату процесін жеңілдетеді-бір адам қысқа уақыт ішінде шкафта телекоммуникациялық жабдықты орнатып, қажет болған жағдайда жабдықты бөлшектей алады. Сонымен қатар, оған арнайы құралдарды сатып алу немесе өлшемдерін өзгерту қажет емес.

Серверлік сөре немесе телекоммуникациялық шкаф стандартты өлшемдерде қолданылады: ені – 482,6 ММ (19 дюйм) және тереңдігі – 900 мм (35,43 дюйм). Серверлік тіректің салмағы - 30,5 кг. шкафтың биіктігі 2,4 м-ден аспауы керек.телекоммуникациялық шкафтар пассивті жабдықты да, белсенді жабдықты да орнатуға және орнатуға арналған. Серверлік тірекке екі сервер және 4 кг коммутатор орнатылды, бұл жалпы 42,5 кг құрайды.шкафқа төмен ток, электр кабельдерін енгізу төменнен жүзеге асырылады, өйткені бөлмені жобалау кезінде жалған еден қолданылды.

4.2 сурет – Электр және телекоммуникациялық кабельдермен әдеттегі жалған еден

ANSI/TIA/EIA-607-B "коммерциялық ғимараттардың телекоммуникациялық жүйелерінің жерге қосу және электр қосылымдарына қойылатын талаптар" стандартының талаптарын ескере отырып, серверлік шкафтар жерге тұйықталуға жатады, өйткені ток өткізгіш бөліктердің оқшаулауы бұзылған кезде металл конструкциялар адамдар үшін қауіпті кернеуде болуы мүмкін. Жерге қосу телекоммуникациялық шкафтың Корпусы мен басқа да металл конструкциялар арасындағы ықтимал айырмашылықты қауіпсіз мәнге дейін азайтуға, сондай-ақ шкафтың өзінде де, ондағы жабдықта да статикалық электр энергиясының жиналуын болдырмауға мүмкіндік береді.

Серверлік тіректің жерге тұйықталуы, әдетте, 4 мм^2-ден асатын мыс өткізгіштің көмегімен жүзеге асырылады, ол ғимараттағы телекоммуникациялық жерге қосу шинасымен бұрандалы қосылыс арқылы қосылады.

Жобалау құжаттамасын оқуды жеңілдету, қандай да бір штаттан тыс жағдайларда жылдам бағдарлау немесе қатесіз монтаждау жұмыстарын орындау үшін әрбір пассивті элемент (розеткалар, кәбілдер, кростық панельдер, коммутациялық жабдық, кәбіл-арналар немесе жерге қосу элементтері), тіпті телекоммуникациялық үй-жайдың өзінің бірегей сәйкестендіргіші және таңбасы болуы тиіс. Бұл талап ANSI/TIA/EIA-606A "коммерциялық ғимараттардың телекоммуникациялық инфрақұрылымын басқару"стандартында қарастырылған.

Сондай-ақ, ANSI/TIA/EIA-942 "жылыту, желдету және кондиционерлеу" пункті бар телекоммуникациялық деректер орталығының құрылымы "стандартының талаптарына сәйкес серверлік бөлмеде талаптарға жауап беретін тиімді желдету және ауаны баптау жүйесін құру қажет. Бұл жүйенің дұрыс конфигурациясы жабық бөлмедегі ауаның барлық параметрлерін (температура, ауа ағынының жылдамдығы, тазалық, ылғалдылық) автоматты түрде ұстап тұруға мүмкіндік береді. Бірнеше телекоммуникациялық шкафтарды орналастырған кезде техникалық жабдықты тиімді салқындату үшін "суық" және "ыстық" ауа дәліздерін көздеу қажет.

Есептеу кезінде өндірістік үй-жайлардағы ауаны баптау жүйесін есептеу әдісі қолданылды.

4.2 Температура айырмашылығына байланысты жылу ағынын есептеу

Жазғы кезеңде үй-жайдың сыртқы қоршаулары арқылы жылудың түсуі күн сәулесінің әсерінен тәулік ішінде сыртқы ауа температурасының және жылу ағынының айтарлықтай ауытқуына байланысты болады.

Алайда, қыста күндізгі сыртқы температура мен жылу ағынының ауытқуы нашар көрінеді, сондықтан есептеулер стационарлық режимнің болжамынан жасалады.

Жылы кезең үшін сыртқы ауаның есептелген температурасы ($t_{Нрасч}$) ең ыстық айдың орташа температурасы 13 сағатқа, ал суық кезең үшін ең суық

айдың орташа температурасы 13 сағатқа сәйкес келеді. Ішкі ауаның есептік температурасы ($t_{Врасч}$) менің жағдайымда технологиялық талаптарды ескере отырып таңдалады, өйткені қызмет көрсетуші персонал серверлік үй-жайда өте сирек болады.

Сыртқы және ішкі ауаның есептік температурасы ҚР ҚНжЕ 2.04-01-2001 алынады.

$Q_{огр}$ жылу мөлшері келесі формула бойынша анықталады (4.1):

$$Q_{огр} = V_{пом} \cdot X_0 \cdot (t_{Нрасч} \cdot t_{Врасч}) \text{ (Вт)} \quad (4.1)$$

мұндағы, $V_{пом}$ – бөлме көлемі, м3;

X_0 – меншікті жылу сипаттамасы, Вт/м3*°С (X_0=0,42 Вт/м3*°С);

$t_{Нрасч}$= 27,6 °С, tВрасч = 22 °С – жылы мезгіл үшін;

$t_{Врасч}$ = -10 °С, tВрасч = 19 °С – суық мезгіл үшін.

Жылы мезгіл үшін жылу мөлшері қайдан келеді:

$$Q_{огр} = 45 \cdot 0.42 \cdot (27.6 - 22) = 105.84 \text{ (Вт)}$$

Суық мезгіл үшін:

$$Q_{огр} = 45 \cdot 0.42 \cdot (-10 - 19) = -548.1 \text{ (Вт)}$$

Өйткені серверлік үй-жайда жоқ терезе ұялары болса, онда қажеттілігін есептеу терезе арқылы жылудың түсуі болмай қалған.

4.3 Жылулық пайда жылғы жарық беру аспаптарын және жабдықты

Бөлмеде жарықсыз жасай алмайтыны анық, сондықтан жарықтандыру кезінде пайда болатын жылуды анықтау қажет. Шамдардың жұмысы кезінде бөлінетін жылу мөлшері келесі формула бойынша анықталады (4.2):

$$Q_{осв} = \eta \cdot N_{осв} \cdot S_{пом} \quad (4.2)$$

мұндағы, η – электр энергиясының жылу энергиясына ауысу коэффициенті;

$N_{ОСВ}$ – шамдардың орнатылған қуаты, Вт/м^2;

$S_{пом}$ – бөлменің ауданы, м^2.

Жақсы жарықтандырылған бөлмелер үшін $N_{ОСВ}$ 50-ден 100 Вт/м^2-ге дейін қабылданады.

Флуоресцентті лампаларды қолданғанда η 0,5-тен 0,6-ға дейін қабылданады. Бір шамның қуаты 60 Вт, $N_{ОСВ}$ = 75 Вт/м^2 құрайды. Осы деректерді негізге ала отырып:

$$Q_{осв} = 0,55 \cdot 60 \cdot 75 = 2\ 475\ Вт$$

Бөлмедегі жабдық ретінде Nutanix NX-3050 серверлері серверлік сөреде және техникалық сипаттамаларға сүйене отырып, орташа есеппен 1100 Вт ($Q_{орг}$) тұтынады.

4.4 Жалпы жылу балансы және сплит-жүйе кондиционерін таңдау

Жоғарыда келтірілген есептеулер мен алынған мәліметтерден біз барлық жылу ағындарының қосындысын формула бойынша анықтаймыз (4.3):

$$Q_{общ} = Q_{огр} + Q_{осв} + Q_{орг} \quad (4.3)$$

Мәндерді қоямыз:

$$Q_{общ} = 105,84 + 2\ 475 + 2 \cdot 1100 = 4\ 680,84\ Вт$$

Бөлмедегі жылу шығынының есептелген мәніне сәйкес біз оның салқындату қабілетіне назар аудара отырып, сплит жүйесінің кондиционерін таңдаймыз. Кондиционерді оның суық өнімділігінің мәні қорды ескере отырып, жылу түсімдерінің мәніне барынша жақын болатындай етіп таңдау қажет.

Mitsubishi Electric MSZ-GF VE кондиционері талаптарға сай келеді (4.3-суретті қараңыз), оның сипаттамалары 4.1-кестеде келтірілген:

4.1 кесте – Таңдалған кондиционердің сипаттамалары

Өндіруші	Mitsubishi Electric
Суық өнімділігі	7100 Вт
Қуатты тұтыну	2130 Вт
Маусымдық энергия тиімділігі (SEER)	6.80 (A++)
Жылу өнімділігі	8100 Вт
Қуатты тұтыну (жылыту)	2230 Вт
Маусымдық энергия тиімділігі (SCOP)	4.20 (A+)
Ауа шығыны (мин.)	852 $м^3$/ч
Ауа шығыны (макс.)	1068 $м^3$/ч
Шу деңгейі (мин.)	30 дБ
Шу деңгейі (макс.)	49 дБ
Салмағы	16 кг
Өлшемдері ЕхТхБ	1100х238х325 $мм^3$
Қуат кернеуі: В, ф, Гц	220-240 В, 1 ф, 50 Гц
Макс. жұмыс тогы	16,6 А
Құбыр диаметрі (сұйық)	9,52 (3/8)
Құбыр диаметрі (газ)	15,88 (5/8)
Дренаж диаметрі	16

4.1 кестенің жалғасы

Өндіруші	Mitsubishi Electric
Магистральдың максималды ұзындығы	30 м
Максималды биіктік айырмашылығы	15 м
Сыртқы температураның кепілдендірілген ауқымы: салқындату	-10 … +46 °C
Сыртқы температураның кепілдендірілген ауқымы: жылыту	-15 … +24 °C WB
Өндіруші ел	Таиланд
Сыртқы блокпен жиынтықта қолданылады	MUZ-GF71VE-ER1

4.3 сурет – Mitsubishi Electric MSZ-GF VE ішкі блогы

5 Жобаның техникалық-экономикалық негіздемесі

5.1 Жұмыстың сипаттамасы және техникалық-экономикалық негіздемесі

Дипломдық жобаның тақырыбы - "желілік трафикті жинау және терең талдау технологиялары". Дипломдық жобаның міндеті-дестетерді берудің жоғары жылдамдықтарында трафикті түсіру және оны жіктеу мәселелерін шешуге бағытталған қолданыстағы ашық бағдарламалық жасақтаманы талдау. Бұл бөлімде бағдарламалық қамтамасыз етуді әзірлеудің экономикалық құрамдас бөлігі ұсынылған, Еңбек және қаржы шығындары айқындалған, жұмыстың экономикалық тиімділігінің көрсеткіштері есептелген.

5.2 Бағдарламалық жасақтаманы әзірлеудің көлемі мен күрделілігін анықтау

Бағдарламалық жасақтаманың құнын бағалау және дамудың экономикалық әсерін анықтау шығындар сметасын есептеуді білдіреді.

Бағдарламалық жасақтаманы әзірлеу шығындарының сметасын есептеудің негізі бағдарламалық өнімнің көлемі болып табылады. БҚ жалпы көлемі (V_o) бағдарламада (5.1) формула бойынша іске асырылатын функциялардың саны мен көлеміне қарай айқындалады):

$$V_o = \sum_{i=1}^{n} V_i \qquad (5.1)$$

мұндағы, V_i – БҚ жеке функцияның көлемі, LOC (Lines Of Code);

n – функциялардың жалпы саны.

(5.1) формуласы бойынша бағдарламалық өнімнің жалпы көлемін табамыз:

$$V_o = 11\ 238 \text{ LOC}$$

Бағдарламалық өнімнің көлемін (бастапқы код жолдарының санын) есептеу БҚ-ның нақты түрін (А Қосымшасы) айқындауды, БҚ-ның барлық функцияларының көлемін техникалық сипаттауды және табуды көздейді. Дипломдық жобаның техникалық-экономикалық негіздемесі қолданыстағы нормативтерге (Б қосымшасы) немесе бұрын іске асырылған ұқсас жобаларға негізделген болжамды бағаларды ұсынады.

Бағдарламалық өнімнің біз тапқан түрдегі жалпы көлемі есептеулерде пайдаланылмайды, оның орнына формула бойынша бағдарламалық қамтамасыз етудің нақтыланған көлемі қолданылады:

$$V_y = \sum_{i=1}^{n} V_{yi} \qquad (5.1)$$

мұндағы, V_{yi} – бағдарламалық өнімнің жеке функциясының нақтыланған көлемі, LOC.

5.2 формуласы бойынша БҚ нақтыланған көлемі:

$$V_y = 12\ 000\ LOC$$

Шағын жобалар үшін жалпы еңбек сыйымдылығы (5.3) формула бойынша есептеледі:

$$T_o = T_н \cdot K_c \cdot K_т \cdot K_н \qquad (5.3)$$

мұндағы, $T_н$ – бағдарламалық жасақтаманы әзірлеудің нормативтік күрделілігі;

K_c – БҚ бойынша күрделілік коэффициенті (5.1 кесте);

$K_т$ – стандартты кітапханаларды, модульдерді әзірлеу кезінде пайдалану дәрежесінің түзету коэффициенті (5.2-кесте);

$K_н$ – БҚ жаңалығы дәрежесінің коэффициенті (5.3-кесте).

5.1 кесте – Қосымша еңбек шығындарының күрделілік коэффициенті

БҚ сипаттамалары	K_c мәндері
Кеңейтілген операциялық ортада БҚ жұмыс істеуі (басқа БҚ-мен байланыс)	0,08
Интерактивті қол жетімділік	0,06
Күрделі құрылымдарда деректерді сақтауды, жүргізуді және іздестіруді қамтамасыз ету	0,07
БҚ бір мезгілде бірнеше сипаттамалардың болуы	
2 сипаттамалары 3 сипаттамалары 3-тен көп сипаттама	0,12 0,18 0,26

Әзірленген бағдарламалық жасақтамада үш сипаттамадан көп болғандықтан, күрделілік коэффициенті 0,26 құрайды.

5.2 кесте – Үлгілік бағдарламалар мен БҚ ($K_т$) стандартты модульдерін пайдалануды ескеретін түзету коэффициентінің мәндері)

БҚ әзірлейтін іске асырылатын функцияларды стандартты модульдермен, үлгілік бағдарламалармен және т. б. қамту дәрежесі	$K_т$ мәндері
60 % және одан жоғары	0,6
40 %-дан 60%-ға дейін	0,7
20 %-дан 40 %-ға дейін	0,8
20 %-ға дейін	0,9
БҚ әзірлеуде стандартты кітапханалар мен модульдер пайдаланылмайды	1,0

Әзірленген бағдарламалық өнімде стандартты кітапханалар мен модульдерді пайдалану дәрежесі 60%-дан жоғары, сәйкесінше түзету коэффициенті 0,6-ға тең.

5.3 кесте – Бағдарламалық өнім жаңалығының түзету коэффициенттері

Жаңалық санаты	Жаңалық дәрежесі	Пайдаланылуы		$К_н$ мәні
		ДК жаңа түріне негізделген	Жаңа ОЖ ортасында	
В	ДК және ОЖ конфигурациясының бұрын игерілген түрлері үшін әзірленген БҚ-ның белгілі бір параметрлік қатарының дамуы болып табылатын БҚ	-	-	0,7

Әзірленген бағдарламалық өнімнің жаңалық санаты "В", сәйкесінше жаңалық дәрежесінің коэффициенті 0,7 құрайды.

Бағдарламалық қамтамасыз етуді әзірлеудің нормативтік еңбек сыйымдылығы есептеуге қабылданған бағдарламалық өнімнің нақтыланған көлемі мен күрделілік санаты негізінде айқындалады. Осы жобаның Нормативтік еңбек сыйымдылығы 260-қа тең (В қосымшасы), яғни жұмыс күрделіліктің үшінші дәрежесіне және бастапқы кодтың 12 000 жолынан көлемге ие.

Бұрын алынған деректерді пайдалана отырып, (5.3) формула бойынша жобаны әзірлеудің жалпы еңбек сыйымдылығын есептейміз:

$$Т_о = 260 \cdot 0{,}26 \cdot 0{,}6 \cdot 0{,}7 = 28{,}392 \text{ (адам/күн)}$$

Әзірлеушілердің жоспарланған саны біреуіне тең, біз бағдарламалық өнімді іске асыру үшін қажетті жоспарланған мерзімдерді формула бойынша есептейміз (5.4):

$$Т_р = \frac{Т_о}{Ч_р \cdot Ф_{эф}} \qquad (5.4)$$

мұндағы, $Ф_{эф}$ – бір қызметкердің бір жылдағы, күндегі жұмыс уақытының тиімді қоры;

$Ч_р$ – БҚ бойынша әзірлеушілер саны.

Өз кезегінде жұмыс уақытының тиімді қоры мына формула бойынша анықталады (5.5):

$$Ф_{эф} = Д_г – Д_п – Д_в – Д_о \qquad (5.5)$$

мұндағы, $Д_г$ – жылдағы күндер саны;

$Д_п$ – жылдағы мереке күндерінің саны;

$Д_в$ – бір жылдағы демалыс күндерінің саны;

$Д_о$ – демалыс күндерінің саны.

2018 жылға:

$$Ф_{эф} = 365 – 14 – 118 – 21 = 212 \text{ (дней)}$$

Содан кейін жоспарланған даму мерзімі:

$$Т_р = \frac{28,392}{(1 \cdot 212)} = 0,1339 \text{ жыл} = 49 \text{ (күн)}$$

Орындалған есептеулердің нәтижелері 5.4-кестеде келтірілген:

5.4 кесте – Есептеу нәтижелері

Атауы	Шартты белгі	Мәні
БҚ жалпы көлемі	$V_о$	11 238 LOC
БҚ бойынша нақтыланған көлем	$V_у$	12 000 LOC
Дамудың нормативтік күрделілігі	$Т_н$	260 күн
Жалпы еңбек сыйымдылығы	$Т_о$	28,392 адам/күн
Тиімді жұмыс уақыты қоры	$Ф_{эф}$	212 күн
Жобаны әзірлеу мерзімі	$Т_р$	49 күн
Жобаны орындаушылар саны	$Ч_р$	1 адам

5.3 Ақпараттық технологияларды әзірлеуге арналған шығындарды есептеу

Ақпараттық технологиялар ($С_{пі}$) түріндегі жобалық шешімді әзірлеуге арналған толық шығындарды есептеу (5.6):

$$С_{пі} = Р_{фот} + З_{зсі} + М_і + Р_{сі} + А_і + П_{зі} + Р_{ні} \quad (5.6)$$

мұндағы, $З_{фот}$ – әзірлеушілердің жалпы еңбекақы қоры, тг;

$З_{сзі}$ – әлеуметтік салық бойынша аударымдар, тг;

$М_і$ – материалдарға арналған шығындар, тг;

$Р_{сі}$ – жобалық шешімді әзірлеу үшін қажетті арнайы бағдарламалық құралдарға арналған шығындар, тг;

$А_і$ – техника амортизациясына байланысты шығындар, тенге;

$П_{зі}$ – басқа шығындар, тг;

$Р_{ні}$ – үстеме шығындар, тг.

Әзірлеушілердің еңбекақы қорының мөлшері (5.7) формула бойынша есептеледі:

$$З_{фот} = З_{оі} + З_{ді} \quad (5.7)$$

мұндағы, $З_{оі}$ – негізгі жалақы, тенге;

$З_{ді}$ – қосымша жалақы, тенге.

Белгілі бір бағдарламалық өнімге орындаушылардың негізгі жалақысы мына формула бойынша есептеледі (5.8):

$$З_{оі} = \sum_{i=1}^{n} Т_ч \cdot Т_ч \cdot Ф_п \cdot К \quad (5.8)$$

мұндағы, n – БҚ әзірлеумен айналысатын орындаушылар саны;

$Т_{чі}$ – і-ші Орындаушының сағаттық тарифтік мөлшерл, тг;

$Ф_п$ – і орындаушының жұмыс уақытының жоспарлы қоры, күндер;

$Т_ч$ – күніне жұмыс сағаттарының саны, сағат;

К – сыйақы коэффициенті.

ҚР Еңбек Кодексі бойынша күніне Еңбек жұмыс сағаттарының саны 8-ге тең. Жұмыс уақытының жоспарлы қоры 49 күнге тең. Сыйақы коэффициенті-1.2. Орындалатын жұмыстардың түрі мен күрделілігі туралы деректер бойынша нақты БҚ әзірлеуге қатысатын мамандар тобының кестесі жасалады (5.5-кесте).

5.5 кесте – БҚ әзірлеуге қатысатын орындаушы мамандар тобының штат кестесі

Маман-орындаушы	Саны, адам	Айына жалақы, тенге
Бағдарламашы	1	150 000
Жалпы		150 000

Сағаттық тарифтік мөлшерлеме мынадай формула бойынша есептеледі (5.9):

$$Т_ч = \frac{Т_м}{Ф_р} \quad (5.9)$$

мұндағы, $Т_м$ – айлық тарифтік мөлшерлеме, тг;

$Ф_р$ – жұмыс уақытының орташа айлық нормасы, сағат.

Айлық тарифтік мөлшерлеме (5.10):

$$Т_м = Т_{м1} \cdot Т_к \quad (5.10)$$

мұндағы, $Т_{м1}$ – 1-разрядтағы қолданыстағы айлық тарифтік мөлшерлеме;

$Т_к$ – біздің жағдайда 1-ге тең тарифтік коэффициент.

Сонда:

$$Т_м = 150\,000 \cdot 1 = 150\,000 \text{ (тг)}$$

Жұмыс уақытының орташа айлық нормасы бір айдағы жұмыс күндерінің санына (22 күн) күніне жұмыс сағаттарының (8 сағат) нормасының туындысы болып табылады. (5.9) формуласы бойынша орындаушының сағаттық тарифтік мөлшерлемесі:

$$Т_ч = \frac{150\,000}{176} = 852 \text{ (тг/сағат)}$$

Қосымша жалақы негізгі жалақының 10% - ын құрайды және мына формула бойынша есептеледі (5.11):

$$З_{ді} = \frac{З_{oi} \cdot Н_д}{100} \qquad (5.11)$$

мұндағы, $Н_д$ – әзірлеушілердің қосымша жалақысының коэффициенті:

$$З_{ді} = 400\,780{,}8 \cdot 0{,}1 = 40\,078{,}08 \text{ (тг)}$$

Әзірлеушінің жалақы қоры қайдан келеді:

$$З_{фот} = 400\,780{,}8 + 40\,078{,}08 = 440\,858{,}88 \text{ (тг)}$$

Әлеуметтік салық қызметкердің табысынан 9,5% - ды құрайды (485-б., ҚР СК 1-т.) және (5.12) формула бойынша есептеледі):

$$З_{сзі} = (З_{фот} – З_{по}) \cdot 9{,}5\% \qquad (5.12)$$

мұндағы, $З_{по}$ – $З_{ЕТҚ}$-ның 10% - ын құрайтын зейнетақы аударымдарына әлеуметтік салық салынбайды, яғни (5.13) формула бойынша айқындалады:

$$З_{по} = З_{фот} \cdot 10\% \qquad (5.13)$$

(5.12, 5.13) формулаларына сәйкес зейнетақы аударымдары мен әлеуметтік салық сәйкесінше:

$$З_{по} = 440\,858{,}88 \cdot 0{,}1 = 44\,085{,}888 \approx 44\,085{,}89 \text{ (тг)}$$

$$З_{сзі} = (440\,858{,}88 – 44\,085.888) \cdot 0{,}095 = 37\,693{,}43424 \approx 37\,693{,}43 \text{ (тг)}$$

Материалдарға жұмсалатын шығындардың шамасы (5.14):

$$М_i = \frac{З_{осн} \cdot Н_{мз}}{100\%} \qquad (5.14)$$

мұндағы, $Н_{мз}$ – негізгі жалақыдан материалдар шығынының нормасы, 3-5%.

Материалдардың құнын табыңыз:

$$М_i = \frac{400\,780{,}8 \cdot 3}{100} = 12\,023{,}424 \approx 12\,023{,}42 \text{ (тг)}$$

"Арнайы жабдықтау" ($Р_{ci}$) бабы бойынша шығындар бағдарламалық қамтамасыз етуді әзірлеу үшін арнайы мақсаттағы қосалқы техникалық және бағдарламалық құралдарды сатып алуға арналған қаражат шығындарын қамтиды және (5.15) формула бойынша есептеледі):

$$Р_{ci} = \sum_{i=1}^{n} Ц_{ci} \qquad (5.15)$$

мұндағы, $Ц_{ci}$ – нақты арнайы жабдықтың құны, тг;

n – қолданылатын арнайы жабдықтың саны.

Өнімді толыққанды әзірлеу үшін қажетті техникалық құралдар мен БҚ сатып алуға арналған қаражат шығындары 5.6-кестеде келтірілген.

5.6 кесте – Техникалық және бағдарламалық құралдарды сатып алуға арналған шығындар

Атауы	Құны, тг
Lenovo IdeaPad IP110S ноутбугы	72 000
Ritmix тінтуірі сияқты манипулятор	500
Жалпы	72 500

35%-ғы амортизация нормасын ескере отырып, амортизациялық аударымдарды табамыз:

$$А_i = \frac{35 \cdot 72\,500 \cdot 49}{100 \cdot 12 \cdot 30} = 3\,453{,}8 \text{ (тг)}$$

"Өзге шығындар "бабы бойынша шығындар ($П_{зi}$) мынадай формула бойынша арнайы ғылыми-техникалық әдебиетті сатып алуға арналған шығындарды қамтиды (5.16):

$$П_{зi} = \frac{З_{oi} \cdot Н_{пз}}{100\%} \qquad (5.16)$$

мұндағы, $Н_{пз}$ – жалпы ұйым бойынша өзге де шығындар нормативі, 20 %.

Өзге де шығындарға арналған шығындар:

$$П_{зi} = \frac{400\,780{,}8 \cdot 20}{100} = 80\,156{,}16 \text{ (тг)}$$

"Үстеме шығындар" ($Р_{ні}$) бабы бойынша шығындар басқару аппаратын ұстау қажеттілігімен және (5.17) формуласы бойынша жалпы шаруашылық мұқтаждықтарға арналған шығыстармен байланысты:

$$Р_{зі} = \frac{З_{oi} \cdot Н_{пз}}{100} \quad (5.17)$$

мұндағы, $Н_{рн}$ – жалпы ұйым бойынша үстеме шығыстар нормативі, 70%.

Шығындарды "үстеме шығындар" бабы бойынша есептейміз:

$$Р_{зі} = \frac{400\,780{,}8 \cdot 70}{100} = 280\,546{,}56 \text{ (тг)}$$

Осылайша, алынған мәліметтерге сүйене отырып, бағдарламалық жасақтаманы әзірлеу құны:

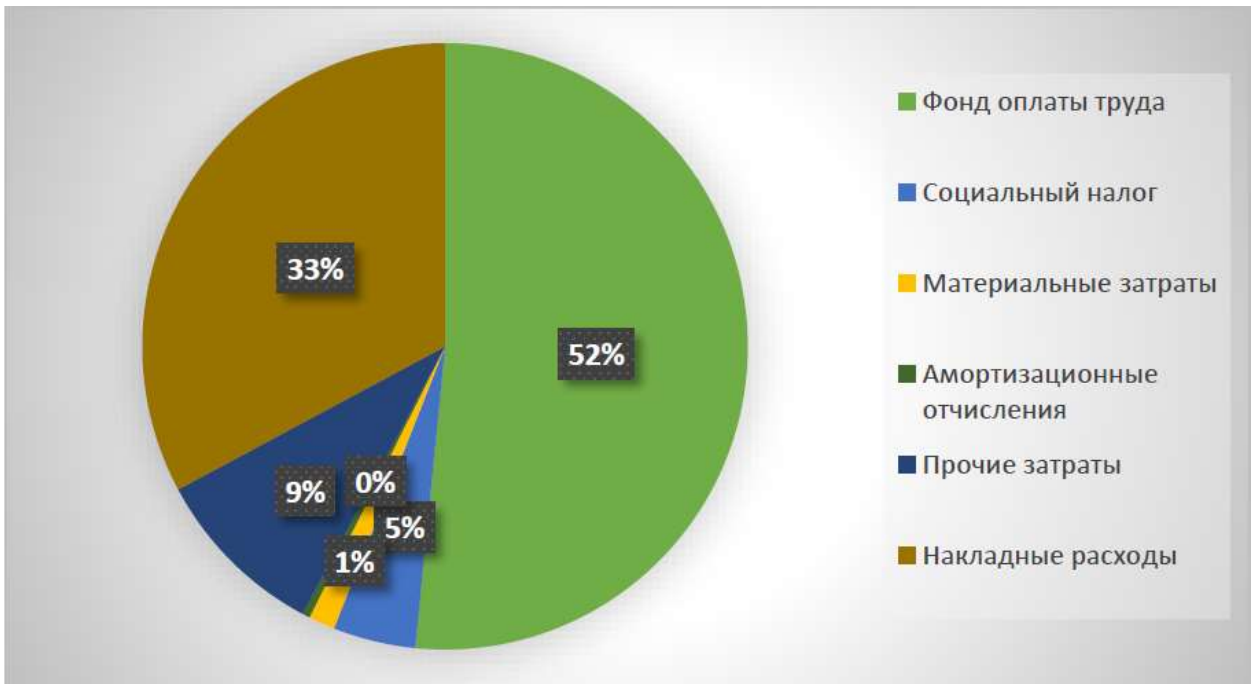
$$С_{пі} = 440\,858{,}89 + 37\,693{,}43 + 12\,023{,}42 + 72\,500 + 3\,453{,}8 + 80\,156{,}16 + \\ + 280\,546{,}56 = 927\,232{,}26$$

$$С_{пі} = 927\,232{,}26 \text{ (тг)}$$

5.7-кестеде жобаға есептелген шығыстар жинақталған, 5.1-суретте шығыстардың құрылымы көрсетілген.

5.7 кесте – Жобаға арналған шығындар

Әзірлеу шығындары	Шартты белгі	Мәні	Жалпы сомадан пайызбен
Еңбекақы төлеу қоры	$З_{фот}$	440 858,89	51,58
Әлеуметтік салық	$З_{сзі}$	37 693.43	4,41
Материалдар	$М_i$	12 023,42	1,41
Амортизация	$А_i$	3 453,8	0,40
Басқа шығындар	$П_{зі}$	80 156,16	9,38
Үстеме шығындар	$Р_{ні}$	280 546,56	32,82
Жалпы		854 732,26	100%



5.1 сурет – Жобаға арналған шығыстардың құрылымы

Жобаны іске асырудағы негізгі шығындар қызметкердің жалақысын құрады, алынған мәліметтерге сүйене отырып, біз бағдарламалық өнімнің жалпы бағасын есептейміз.

5.4 Бағдарламалық өнім бағасын есептеу

Әзірленетін бағдарламалық қамтылымның бағасын есептеу (5.18) формула бойынша көбейту қажеттілігінсіз бір ұйымның екіншісіне бағдарламалық өнімге тапсырыс беруіне сүйене отырып жүргізіледі:

$$Ц_{пп} = З_{рпп} + П_{п} + НДС \quad (5.18)$$

мұндағы, $Ц_{пп}$ – бағдарламалық өнімнің бағасы, тг;

$З_{рпп}$ – бағдарламалық өнімді әзірлеу шығындары, тг;

НДС – қосылған құн салығы, тг.

Жоспарланған пайда (5.19) формула бойынша есептеледі:

$$П_{п} = З_{рпр} \cdot R_{нпп} \quad (5.19)$$

мұндағы, $R_{нпп}$ – ұйым айқындайтын бағдарламалық өнімнің нормативтік рентабельділігі.

Бағдарламалық өнімге есептелген НДС келесі формула бойынша анықталады (5.20):

$$НДС = (З_{рпр} + П_{п}) \cdot k_{НДС} \quad (5.20)$$

мұндағы, $k_{НДС}$ – қосылған құн салығының ставкасы.

(5.19, 5.20) формулалары бойынша Зрпп = Спі, Rнпп = 22 % екенін ескере отырып, жоспарланған пайда мен НДС есептейміз:

$$П_п = 927\ 232{,}26 \cdot 0{,}22 = 203\ 991{,}0972 \approx 203\ 991{,}10 \text{ (тг)}$$

$$НДС = (927\ 232{,}26 + 203\ 991{,}10) \cdot 0{,}12 = 135\ 746{,}8032 \approx 135\ 746{,}8 \text{ (тг)}$$

Содан кейін сату бағасы:

$$Ц_{пп} = 927\ 232{,}26 + 203\ 991{,}10 + 135\ 746{,}8 = 1\ 226\ 970{,}16 \text{ (тг)}$$

Кәсіпорын үшін әзірленген жүйені енгізу шығындары бір реттік болып табылады және осы жобаға инвестициялардың жалпы құнын құрайды.

Күрделі салымдар (5.21) формуласы бойынша болады:

$$К = З_{КТС} \cdot (1 + к_{ТУН}) + З_{ПО} + З_{ФОТВ} + З_{ОФР} + Р_Н \qquad (5.21)$$

мұндағы, $З_{КТС}$ – техникалық құралдар кешенін сатып алуға арналған шығындар, тг (5.6 кесте);

$k_{ТУН}$ – техникалық құралдар кешенін тасымалдау, орнату және баптау коэффициенті техникалық құралдар құнының 10%-ына тең;

$З_{ПО}$ – БҚ сатып алуға арналған шығындар, тг (0-ге тең, себебі шешімдердің open source пайдаланылады);

$З_{ФОТВ}$ – қызметкерлерге еңбекақы төлеуге арналған шығындар, тг (5.7 кесте);

$З_{ОФР}$ – қызметкерлердің жалақысынан әлеуметтік салық, тг (5.7 кесте);

$Р_Н$ – үстеме шығындар, тг (5.7 кесте).

Сонда күрделі салымдар:

$$К = 72\ 500 \cdot (1 + 0) + 0 + 440\ 858{,}89 + 37\ 693{,}43 + 280\ 546{,}56 = 831\ 598{,}88 \text{ (тг)}$$

Бағдарламалық өнімді сатудың және капиталды инвестициялаудың қорытынды бағасының мәліметтерін қолдана отырып, экономикалық тиімділік көрсеткіштері бар.

5.5 Экономикалық тиімділік көрсеткіштерін есептеу

Күрделі салымдардың экономикалық тиімділігінің нормативтік коэффициенті мына формула бойынша анықталады (5.22):

$$Е_н = \frac{1}{Т_н} \qquad (5.22)$$

мұндағы, $Т_н$ – күрделі салымдардың өтелуінің нормативтік мерзімі, жыл.

$T_н$ – бағдарламалық өнімдер үшін өтелу мерзімі 4 жылға тең болады, содан:

$$E_н = \frac{1}{4} = 0,25$$

Күрделі салымдардың экономикалық тиімділігінің есептік коэффициенті мына формула бойынша анықталады (5.23):

$$E_р = \frac{1}{T_р} \quad (5.23)$$

мұндағы, $T_р$ – күрделі салымдардың өтелу мерзімі, жыл.

Күрделі салымдардың өтелімділігінің есептік мерзімі (5.24) формуласы бойынша:

$$T_р = \frac{К}{Ц_{пп}} \quad (5.23)$$

$$T_р = \frac{831\ 598,88}{1\ 266\ 970,16} \approx 0,66$$

Қайдан:

$$E_р = \frac{1}{0,66} \approx 1,52$$

Тиімділіктің есептелген экономикалық көрсеткіштері 5.8-кестеде келтірілген.

5.8 кесте – Экономикалық тиімділік көрсеткіштері

Параметрі	Мәні
Жоспарланған пайда	203 991,10 тг
НДС	135 746,80 тг
НДС есебімен сату бағасы	1 266 97016 тг
Күрделі салымдар	831 598,88 тг
Өтелу мерзімі	0,6 жыл
Экономикалық тиімділік коэффициенті	1,52

Осылайша, күрделі салымдардың экономикалық тиімділігінің есептік коэффициенті нормативтіктен көп ($E_р > E_н$), бұл жобаның өтелімділігі 4 жылға (0,66 жыл) қарағанда аз екенін көрсетеді.

Экономикалық бөлім бойынша қорытынды жасай отырып, бағдарламалық қамтамасыз етуді әзірлеуге жұмсалған шығындар 927 232,26 теңгені құрайды. 5.7-кестенің деректеріне сүйене отырып, шығыстардың

негізгі бөлігін еңбекақы төлеу қорына арналған шығындар 51,58% және үстеме шығыстар 32,82% құрайды. Бұл бағдарламалық өнімді әзірлеу кезінде жеке құралдар пайдаланылды. Бағдарламалық өнімді сату бағасы НДС есебімен 1 266 970.16 теңгені құрайды.

Жоспарларда бұл бағдарламалық жасақтама зерттеу жұмыстарының қажеттіліктері үшін, бұрын қарастырылған желілік трафикті түсіру және талдау технологияларын одан әрі зерттеу үшін пайдаланылатын болады, сондықтан есептеулерде осы өнімді қазіргі уақытта әкелетін пайда ескерілмеген. Бұл дипломдық жоба экономикалық емес, ғылыми қызығушылық тудырады.

Қорытынды

Бүгінгі таңда желілік трафикті талдау өте кең тақырып. "Желілік трафикті талдау" деп біз нақты уақыт режимінде желілік дестетерді жинақтауға, өңдеуге, жіктеуге, бақылауға және өзгертуге мүмкіндік беретін технологиялар мен оларды іске асырудың жиынтық атауын түсінеміз.

Бұл мәселені қарастыру кезінде күрделі факторлардың бірі желілік трафикті талдау құралдарын дамытудың қосарлануы болып табылады: бір жағынан — талдау алгоритмдері мен тәсілдерін дамыту, екінші жағынан — бұл мәселені тиімді шешу үшін бағдарламалық және аппараттық құралдарды дамыту. Өз кезегінде, бұл терминологиядағы шатасуға да, маркетингтік мақсаттар үшін фактілер мен сандарды саналы түрде басқаруға әкеледі.

Бұл дипломдық жоба желілік трафикті жинаудың және терең талдаудың заманауи технологияларын қарастырады. Олардың кейбіреулері (NETMAP, PF_RING, DPDK) желілік дестетерді жоғары жылдамдықта түсіру өнімділігін тестілеуде қолданылды, басқалары (nDPI, libprotoident) желілік трафикті дәл жіктеуде қолданылды. Тестілеу нәтижелері BSD лицензиясы бойынша таратылатын NETMAP және DPDK шеңберлері 10 гигабиттік өткізу қабілеттілігінің максималды дестетік жылдамдығында дестетік түсірілімге қол жеткізе алатындығын көрсетті. nDPI және libprotoident трафигін жіктеуге арналған Ашық шешімдер хаттамаларды тануда жоғары дәлдікке ие, бірақ олар трафик ағындарын бөлуде аздап шикі болып шықты және сәйкесінше жетілдіруді қажет етеді.

Тіршілік әрекетінің қауіпсіздігі бөлігі бойынша тестілеуде пайдаланылған жабдық орналасқан серверлік үй-жай сипатталған, үй-жайдың жылу теңгерімі есептелген және кондиционерлеу жүйесі таңдалған.

Дипломдық жобаның экономикалық бөлігінде әзірленген зерттеу бағдарламалық өнімінің бағалау құны жасалды, орындалған жұмыстардың Еңбек және қаржылық шығындары анықталды. Бағдарламалық қамтамасыз етуді әзірлеуге жұмсалатын жалпы шығындар 927 232,26 теңгені құрады. Жүргізілген жұмыстар тиімділігінің экономикалық көрсеткіштері айқындалды.

Қысқартулар тізімі

DTM – Data Traffic Management
PCRF – Policy and Charging Rules Function
DPI – Deep Packet Inspection
БЖ – Бағдарламалық жасақтама
EPC – Evolved Packet Core
OSS – Operations Support Systems
QoS – Quality of Service
OSI – Open Systems Interconnection
ISP – Internet Service Provider
HTTP – HyperText Transfer Protocol
HTTPS – HyperText Transfer Protocol Secure
SIP – Session Initiation Protocol
RTP – Real-time Transport Protocol
RTSP – Real Time Streaming Protocol
IGMP – Internet Group Management Protocol
CAGR – Compound Annual Growth Rate
USD – United States Dollar
ОПТЖ – Ортақ пайдаланатын телефон желісі
ҚР – Қазақстан Республикасы
QoE – Quality of Experience
P2P – Peer-to-Peer
HTML – HyperText Markup Language
VoIP – Voice over Internet Protocol
ToS – Type of Service
DSCP – Differentiated Services Code Point
ECN – Explicit Congestion Notification
LAN – Local Area Network
VLAN – Virtual Local Area Network
IEEE – Institute of Electrical and Electronics Engineers
MPLS – Multiprotocol Label Switching
UDP – User Datagram Protocol
TCP – Transmission Control Protocol
CDN – Content Delivery Network
GGC – Google Global Cache
MITM – Man In The Middle
SMTP – Simple Mail Transfer Protocol
DNS – Domain Name System
ЖІШЖ – Жедел-іздестіру іс-шараларының функцияларын қамтамасыз етуге арналған техникалық құралдар жүйесі
SPAN – Switched Port Analyzer
PBR – Policy Based Routing
DPDK – Data Plane Development Kit
POSIX – Portable Operating System Interface

BSD – Berkeley Software Distribution
BPF – Berkeley Packet Filter
LSF – Linux Socket Filter
ОЖ – Операциялық жүйе
ZC – Zero Copy
DMA – Direct Memory Access
NIC – Network Interface Controller/Card
MAC – Media Access Control
IP – Internet Protocol
FTP – File Transfer Protocol
POP – Post Office Protocol
IMAP – Internet Message Access Protocol
IPP – Internet Printing Protocol
MDNS – Multicast Domain Name System
NTP – Network Time Protocol
NFS – Network File System
SSDP – Simple Service Discovery Protocol
BGP – Border Gateway Protocol
SNMP – Border Gateway Protocol
XDMCP – X Display Manager Control Protocol
SMB – Server Message Block
DHCP – Dynamic Host Configuration Protocol
TDS – Tabular Data Stream
PACE – Protocol and Application Classification Engine
NBAR – Network-Based Application Recognition
API – Application Programming Interface
FPGA – Field-Programmable Gate Array
TNAPI – Threaded New Application Programming Interface
PMD – Poll Mode Driver
EAL – Environment Abstraction Layer
CPU – Central Processing Unit
NUMA – Non-Uniform Memory Access
RSS – Receive Side Scaling
SFP – Small Form-factor Pluggable
EPEL – Extra Packages for Enterprise Linux
SFD – Start Frame Delimiter
IPG – InterPacket Gap
TSO – TCP Segmentation Offload
UFO – UDP Fragmentation Offload
GSO – Generic Segmentation Offload
GRO – Generic Receive Offload
LRO – Large Send Offload
SSL – Secure Sockets Layer

Пайдаланылған әдебиеттер тізімі

1 Carrier Big Data, Deep Packet Inspection, Analytics, and Data as a Service 2015 - 2020 // [Research and Markets] URL: https://tinyurl.com/ybudt6km (дата обращения 31.12.2017).

2 Опрос «Мобильная связь в Казахстане» // [Официальная страница компании 4Service Group] URL: https://tinyurl.com/y7xwdyht (дата обращения 31.12.2017).

3 Kevin Roebuck. Deep Packet Inspection (DPI): High-Impact Strategies - What You Need to Know: Definitions, Adoptions, Impact, Benefits, Maturity, Vendors // Australia: Emereo Publishing, 2012, ISBN 1283761904 (ISBN13: 978-1-28376-190-1).

4 Oliver M. Heckmann. The Competitive Internet Service Provider: Network Architecture, Interconnection, Traffic Engineering and Network Design. USA, New York: John Wiley & Sons, 2007, ISBN: 978-0-470-03004-2 – 398 pp.

5 Google GGC Program // [Официальная страница GGC] URL: https://peering.google.com/ (дата обращения 01.01.2018).

6 PACKET(7) // [Linux Programmer's Manual] URL: https://tinyurl.com/ovlgsee (дата обращения 01.01.2018).

7 AF_PACKET source code // [Linux kernel source tree] URL: https://tinyurl.com/y8kcav4e (дата обращения 01.01.2018).

8 PF_RING // [ntop - High Performance Network Monitoring Solutions based on Open Source and Commodity Hardware] URL: https://tinyurl.com/y7j3fywm (дата обращения 01.01.2018).

9 PF_RING source code // [High-speed packet processing framework] URL: https://tinyurl.com/q688mvw (дата обращения 01.01.2018).

10 NETMAP // [Luigi Rizzo - Università di Pisa] URL: https://tinyurl.com/6ex353f (дата обращения 01.01.2018).

11 NETMAP source code // [Netmap - a framework for fast packet I/O] URL: https://tinyurl.com/kwskdty (дата обращения 01.01.2018).

12 DPDK // [Data Plane Development Kit] URL: https://dpdk.org/ (дата обращения 01.01.2018).

13 Linux Socket Filtering aka Berkeley Packet Filter (BPF) // [Linux Kernel Documentation] URL: https://tinyurl.com/kzvequv (дата обращения 01.01.2018).

14 Netgraph – graph based kernel networking subsystem // [FreeBSD Kernel Interfaces Manual] URL: https://tinyurl.com/yce8odk8 (дата обращения 01.01.2018).

15 PACKET_MMAP // [Linux Kernel Documentation] URL: https://tinyurl.com/qz4rb8k (дата обращения 01.01.2018).

16 Linux sk_buff // [Linux Foundation Wiki] URL: https://tinyurl.com/ybdq8adz (дата обращения 01.01.2018).

17 PF_RING ZC (Zero Copy) // [ntop - High Performance Network Monitoring Solutions based on Open Source and Commodity Hardware] URL: https://tinyurl.com/ycyvwc38 (дата обращения 01.01.2018).

18 An Overview of Network Traffic Classification Methods // [International Journal on Recent and Innovation Trends in Computing and Communication] URL: https://tinyurl.com/y9zz3lp5 (дата обращения 01.01.2018).

19 nDPI source code // [Open Source Deep Packet Inspection Software Toolkit] URL: https://github.com/ntop/nDPI (дата обращения 01.01.2018).

20 Libprotoident 2.0.12 // [Network traffic classification library that requires minimal application payload] URL: https://tinyurl.com/y7b3yssl (дата обращения 01.01.2018).

21 Libprotoident Supported Protocols // [List of application protocols that libprotoident is capable of identifying] URL: https://tinyurl.com/yac59ys2 (дата обращения 01.01.2018).

22 Cisco’s NBAR2 Protocol Library // [Bulletins for Cisco products] URL: https://tinyurl.com/ycsgr5kn (дата обращения 01.01.2018).

23 PF_RING application to bridge two network interfaces // [The world's leading software development platform · GitHub] URL: https://tinyurl.com/y9mlqdjl (дата обращения 01.01.2018).

24 Telecommunications Pathways and Spaces. ANSI/TIA-569-C-201 APPROVED: MAY 3, 2012 // [TIA Standard] URL: https://tinyurl.com/y8f762fe (дата обращения 01.01.2018).

25 Generic Telecommunications Bonding and Grounding (Earthing) for Customer Premises. ANSI/TIA-607-B-201 APPROVED: AUGUST 26, 2011 // [TIA Standard] URL: https://tinyurl.com/ybpzodpf (дата обращения 01.01.2018).

26 Administration Standard for Commercial Telecommunications Infrastructure. ANSI/TIA/EIA-606-A-2002 Approved: May 16, 2002 // [TIA Standard] URL: https://tinyurl.com/y83cg8kw (дата обращения 01.01.2018).

27 Telecommunications Infrastructure Standard for Data Centers. ANSI/TIA-942 // [TIA Standard] URL: https://tinyurl.com/2e4tnca (дата обращения 01.01.2018).

28 Строительная климатология (СНиП РК 2.04-01-2001) // [Комплекс 2.04 Внутринний климат и защита от вредных воздействий] URL: https://tinyurl.com/y73uvvef (дата обращения 01.01.2018).

29 Т.Е. Хакимжанов. Расчет аспирационных систем. Дипломное проектирование. Для студентов всех форм обучения всех специальностей – Алматы: АИЭС, 2002 г. (дата обращения 01.01.2018).

30 Mitsubishi Electric MSZ-GF VE // [Device Specification] URL: https://tinyurl.com/y8q7vk7w (дата обращения 01.01.2018).

31 З.Д. Еркешева, Г.Ш. Боканова. Методические указания к выполнению экономической части дипломных работ для студентов специальности 5В070400 – Вычислительная техника и программное обеспечение. – Алматы: АУЭС, 2013 – 40 с. (дата обращения 01.01.2018).

32 Кодекс Республики Казахстан от 25 декабря 2017 года № 120-VI «О налогах и других обязательных платежах в бюджет (Налоговый кодекс)» // [ЮРИСТ – комплекс правовой информации (законодательство) Республики Казахстан] URL: https://tinyurl.com/ybbbndff (дата обращения 01.01.2018).

А қосымшасы

NETMAP көмегімен іске асырылған дестетерді қайта бағыттаудың жеңілдетілген бағдарламасы

```
1 nm_desc *rx_if, *tx_if;
2 rx_if = nm_open("netmap:eth0", NULL, 0, NULL);
3 tx_if = nm_open("netmap:eth1", NULL, 0, NULL);
4
5 netmap_ring *rxring, *txring;
6 rxring = NETMAP_RXRING(rx_if->nifp, rx_if->first_rx_ring);
7 txring = NETMAP_TXRING(tx_if->nifp, tx_if->first_tx_ring);
8
9 while (1) {
10    ioctl(rx_if->fd, NIOCRXSYNC, NULL);
11
12    int pkts = MIN(nm_ring_space(rxring), nm_ring_space(txring));
13
14    if (pkts) {
15       int rx = rxring->cur;
16       int tx = txring->cur;
17
18       while (pkts) {
19          netmap_slot *rs = &rxring->slot[rx], *tx = &txring->slot[tx];
20
21          // copy the packet length
22          ts->len = rs->len;
23
24          // switch buffers
25          uint32_t pkt = ts->buf_idx;
26          ts->buf_idx = rs->buf_idx;
27          rs->buf_idx = pkt;
28
29          // report the buffer change
30          ts->flags |= NS_BUF_CHANGED;
31          rs->flags |= NS_BUF_CHANGED;
32
33          rx = nm_ring_next(rxring, rx);
34          tx = nm_ring_next(txring, tx);
35
36          pkts -= 1;
37       }
38
39       rxring->cur = rx;
40       txring->cur = tx;
41
42       ioctl(tx_if->df, NIOCTXSYNC, NULL);
43    }
44 }
```

Б қосымшасы

PF_RING көмегімен іске асырылған дестетерді қайта бағыттаудың жеңілдетілген бағдарламасы

```
1 #define BURST_SZ 32
2
3 pfring_zc_cluster *zc;
4 pfring_zc_queue *inzq, *outzq;
5 pfring_zc_pkt_buff *buff[BURST_SZ];
6
7 int start_forwarder() {
8    zc = pfring_zc_create_cluster(
9        1,      // cluster id
10       1536,   // buffer length
11       0,      // buffer metadata length
12       65537,  // num of buffers
13       ...);
14
15    // prepare buffers and queues
16    for (int i = 0; i < BURST_SZ; ++i)
17       buff[i] = pfring_zc_get_packet_handle(zc);
18    inzq = pfring_zc_open_device(zc, "zc:eth0", rx_only, 0);
19    outzq = pfring_zc_open_device(zc, "zc:eth1", tx_only, 0);
20
21    // start forwarder
22    pthread_create(..., forwarder_thread, ...);
23
24 }
25
26 void* forwarder_thread() {
27    int tx_queue_not_empty = 0;
28
29    while (1) {
30       // receive packets
31       int num_pkts = pfring_zc_recv_pkt_burst(inzq, buff, BURST_SZ. 0);
32
33       if (num_pkts > 0) {
34          // send packets
35          pfring_zc_send_pkt_burst(outzq, buff, num_pkts, 0);
36          tz_queue_not_empty - 1;
37       } else {
38          if (tz_queue_not_empty) {
39             pfring_zc_sync_queue(i->outzq, tx_only);
40             tx_queue_not_empty = 0;
41          }
42       }
43    }
44 }
```

В қосымшасы

DPDK көмегімен іске асырылған дестетерді қайта бағыттаудың жеңілдетілген бағдарламасы. №1 бөлім

```
1 struct rte_mempool* pktbuf_pool = NULL;
2 int rx_port_id = 1, tx_port_id = 0;
3
4 int main(int argc, char **argv) {
5    rte_eal_init(argc, argv);
6
7    pktbuf_pool =
8      rte_mempool_create("mbuf_pool", NB_MBUF,
9                         MBUF_SIZE, 32,
10                         sizeof(struct rte_pktmbuf_pool_private),
11                         rte_pktmbuf_pool_init, NULL,
12                         rte_pktmbuf_init, NULL,
13                         rte_socket_id(), 0);
14
15    rte_pmd_init_all();
16    rte_eal_pci_probe();
17
18    prepare_dev(rx_port_id);
19    prepare_dev(tx_port_id);
20
21    rte_eal_remote_launch(run, NULL, 1);
22    rte_eal_wait_lcore(1);
23
24    return 0;
25 }
26
27 static void prepare_dev(int id) {
28    int sock_id = rte_eth_dev_socket_id(id);
29
30    rte_eth_dev_configure(id, 1, 1, &port_conf);
31    rte_eth_rx_queue_setup(id, 0, 128, sock_id, &rx_conf, pktbuf_pool);
32    rte_eth_tx_queue_setup(id, 0, 512, sock_id, &tx_conf);
33    rte_eth_promiscuous_enable(id);
34 }
```

Г қосымшасы

DPDK көмегімен іске асырылған дестетерді қайта бағыттаудың жеңілдетілген бағдарламасы. №2 бөлім

```
1 #define MAX_PKT_BATCH 32
2
3 void run() {
4    struct rte_mbuf *pkt_burst[MAX_PKT_BURST];
5    struct rte_mbuf *m;
6    struct rte_mbuf *m_table[MAX_PKT_BURST];
7    int len = 0;
8
9    while (1) {
10       int nb_rx = rte_eth_rx_burst(rx_port_id, 0, pkts_burst,
11                                    MAX_PKT_BATCH);
12
13       for (int j = 0; j < nb_rx; j++) {
14          m = pkts_burst[j];
15          m_table[len++] = m;
16
17          // enough pkts to be sent
18          if (len == MAX_PKT_BATCH) {
19             int ret = rte_eth_tx_burst(tx_port_id, 0, m_table,
20                                        MAX_PKT_BATCH);
21
22             if (ret < MAX_PKT_BATCH) {
23                do {
24                   rte_pktmbuf_free(m_table[ret]);
25                } while (++ret < MAX_PKT_BATCH);
26             }
27
28             len = 0;
29          }
30       }
31    }
32 }
```

Д қосымшасы

Trace-cmd утилитасын шығару

```
net_rx_action() {
 ixgbe_poll() {
  ixgbe_clean_rx_irq() {
   ixgbe_fetch_rx_buffer() {
    ... // allocate buffer for packet
   } // returns the buffer containing packet data
   ...
   napi_gro_receive() {
    // generic receive offload
    dev_gro_receive() {
     inet_gro_receive() {
      udp4_gro_receive() {
       udp_gro_receive();
      }
     }
    }
    netif_receive_skb_internal() {
     __netif_receive_skb() {
      __netif_receive_skb_core() {
       ...
       ip_rcv() {
        ...
        ip_rcv_finish() {
         ...
         ip_local_deliver() {
          ip_local_deliver_finish() {
           raw_local_deliver();
           udp_rcv() {
            __udp4_lib_rcv() {
             __udp4_lib_lookup_skb() {
              __udp4_lib_lookup() {
               ...
               // clone skb & deliver
               flush_stack() {
                udp_queue_rcv_skb() {
                 ... // data preparation
                 // deliver UDP packet & check if buffer is full
                 __udp_queue_rcv_skb() {
                  // deliver to socket queue & check for delivery error
                  sock_queue_rcv_skb() {
                   ...
                   _raw_spin_lock_irqsave();
                   // enqueue packet to socket buffer list
                   _raw_spin_unlock_irqrestore();
                   // wake up listeners
                   sock_def_readable() {
                    __wake_up_sync_key() {
                     _raw_spin_lock_irqsave();
                     __wake_up_common() {
                      ep_poll_callback() {
                       ...
                       _raw_spin_unlock_irqrestore();
                      }
                     }
                     _raw_spin_unlock_irqrestore();
                    }
...
```