REPUBLIKA E SHQIPËRISË
UNIVERSITETI POLITEKNIK I TIRANËS
FAKULTETI I TEKNOLOGJISË SË INFORMACIONIT
DEPARTAMENTI I ELEKTRONIKËS DHE TELEKOMUNIKACIONIT

LUAN RUÇI

PËR MARRJEN E GRADËS

"DOKTOR"

NË "TEKNOLOGJITË E INFORMACIONIT DHE KOMUNIKIMIT"
DREJTIMI "TELEKOMUNIKACION DHE INXHINIERI INFORMACIONI"

DISERTACION

Përmirësimi i performancës energjitike në telefonat inteligjente

Udhëheqës Shkencor

Prof.As. Olimpjon Shurdi

Tiranë, 2020

PËRMIRËSIMI I PERFORMANCËS ENERGJITIKE NË TELEFONAT INTELIGJENTE

Disertacioni

I paraqitur në Universitetin Politeknik të Tiranës

për marrjen e gradës

"Doktor"

në

"Teknologjitë e Informacionit dhe Komunikimit"

drejtimi Telekomunikacion dhe Inxhinieri Informacioni

nga

z. Luan Ruçi

2020

JURIA PËR VLERËSIMIN E DISERTACIONIT PËR FITIMIN E GRADËS SHKENCORE "DOKTOR"

Miraturar

Me vendimin e Këshillit të Profesorëve të FTI-së Nr_____ , datë_____

Kryetari i Jurisë: Prof.Dr Aleksander Xhuvani

Anëtar i Jurisë: Prof.Dr Luan Karçanaj

Anëtar i Jurisë: Prof.Dr Fatmir Hoxha

Anëtar i Jurisë: Prof.Asoc Bexhet Kamo

Anëtar i Jurisë: Prof.Asoc Indrit Enesi

Dekani i Fakultetit të Teknologjisë së Informacionit

Prof.Asoc Vladi Koliçi

Përmbajtja

Lista e Figurave

Lista e Tabelave

Shkurtesat

8PSK	*8 Phase Shift Keying*
3G	*3rd Generation*
3GPP	*3rd Generation Partnership Project*
AAL2	*ATM Asynchronous Transport Mode) Adaptation Layer 2*
ACK	*Acknowledgement*
AP	*Access Point*
BCCH	*Broadcast Control Channel*
BCH	*Broadcast Channel*
BS	*Base Station*
CCCH	*Common Control Channel*
CN	*Core Network*
CPU	*Central Processing Unit*
CQI	*Channel Quality Indicator*
CSD	*Circuit Switched Data*
CTCH	*Common Traffic Channel*
DCCH	*Dedicated Control Channel*
DCH	*Dedicated Channel*
DL	*Downlink*
DRX	*Discontinuous Reception*
DSCH	*Downlink Shared Channel*
DSP	*Digital signal processor*
DTCH	*Dedicated Traffic Channel*
EDGE	*Enhanced Data Rates for GSM Evolution*
EGPRS	*Enhanced Data Rates for GSM Evolution*
EPC	*Evolved Packet Core*
GGSN	*Gateway GPRS Support Node*
GMSC	*Gateway MSC*
GPRS	*General Packet Radio Service*
GPS	*Global Positioning System*
GPU	*Graphichal processing Unit*
GSM	*Global System for Wireless communications*
GTP-U	*GPRS (General Packet Radio System) Tunnelling Protocol for the user plane*
FACH	*Forward Access Channel*
HSDPA	*High-Speed Downlink Packet Access*
HSPA	*High-Speed Packet Access*
LCD	*Liquid Crystal Display*
LTE	*Long Term Evolution*
MAC	*Medium Access Control*
ME	*Mobile Equipment*
MSC	*Mobile Switching Center*
MTP3b	*Message Transfer Part Layer 3 broadband*
NAS	*Non Access Stratum*

NIC	*Network Interface Card*
ICMP	*Internet Control Message Protocol*
IDT	*Inter-Departure Time*
IEEE	*Institute of Electrical and Electronics Engineers*
IETF	*Internet Engineering Task Force*
IP	*Internet Protocol*
OS	*Operating System*
OSI	*Open System Interconnection*
P2P	*Peer-to-Peer*
PCCH	*Paging Control Channel*
PCH	*Paging Channel*
PDCP	*Packet Data Convergence Protocol*
PDP	*Packet Data Protocol*
PHY	*Physical Layer*
PICH	*Paging Indicator Channel*
PPP	*Point-to-Point*
PS	*Packet Switched*
QoS	*Quality of Service*
QoE	*Quality of Experience*
RANAP	*Radio Access Network Application Protocol*
RACH	*Random Access Channel*
RAM	*Random-Access Memory*
RLC	*Radio Link Controller*
RNC	*Radio Network Controller*
RRC	*Radio Resource Control*
RTP	*Real-Time transport*
RTT	*Round Trip Time*
SCCP	*Signaling Control Connection Part*
SDRAM	*Synchronous Dynamic Random Access Memory*
SGSN	*Serving GPRS Support Node*
SoC	*System on Chip*
SN	*Social Networks*
SMS	*Short Message Service*
SIM	*Subscriber Identity Module*
SNR	*Signal-to-Noise Ratio*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
UE	*User Equipment*
UL	*Uplink*
UMTS	*Universal Wireless Telecommunications System*
URA	*UTRAN Registration Area*
USB	*Universal Serial Bus*
USIM	*UMTS Subscriber Identity Module*
UTRA	*UMTS Terrestrial Radio Access*
UTRAN	*UMTS Terrestrial Radio Access Network*
WCDMA	*Wideband Code Division Multiple Access*

Mirënjohje

Dëshiroj të shpreh falenderimet e mia të përzemërta për të gjithë ata që në një mënyrë apo një tjetër dhanë ndihmesën e tyre për ta bërë punën time me doktoraturën më të lehtë dhe më të këndshme. Fillimisht, dëshiroj t'i drejtoj falenderimet e mia të sinqerta udhëheqësit tim shkencor Prof. Asoc. Olimpjon Shurdi tek i cili kam gjetur gjithmonë sugjerime me vlerë, mbështetje dhe orientim shkencor dhe i cili më ka shtyrë gjithmonë drejt kërkimeve dhe ideve të reja ne lidhje me punimin e disertacionit.

Dua të shpreh mirënjohjen time për drejtuesit e Departamentit të Elektronikës dhe Telekomunikacionit në Universitetin Politeknik të Tiranës, të cilët më dhanë mundësinë të vazhdoj studimet e doktoraturës për thellimin e studimeve në fushën e komunikimeve mobile. Njëkohësisht, mirënjohja ime shkon për të gjithë kolegët e Departamentit të Elektronikës dhe Telekomunikacionit për rekomandimet e tyre të vlefshme dhe të pa kursyera sa herë që kam pasur nevojë.

Së fundmi, një falenderim i veçantë i shkon familjes sime e cila ka qenë motivimi dhe shtysa kryesore gjatë gjithë përpjekjeve të mia për përfundimin me sukses të punimit të doktoraturës.

Abstrakt

Pajisjet mobile në komunikimet celulare kanë evoluar me shpejtësi vitet e fundit dhe përdorimi e performanca e tyre janë duke i shtyrë teknologjitë bashkëkohore të baterive në limitet e tyre. Ndrydhja e "burimeve" të baterive ka ardhur si rrjedhojë e ndryshimeve të shpejta në teknologjitë HW, SW si dhe Aplikimeve në smartphonë-t e sotëm. Investigimet në lidhje me eficencën e energjisë tregojnë se rrjetat e radio aksesit janë ndër kontribuesit kryesor në energjinë totale të konsumit të energjisë. Por me kompleksitetin në strukturën HW e SW të smartphoneve të sotëm e bën të domosdoshme evidentimin dhe llogaritjen e konsumit në çdo komponent HW,SW apo tip komunikimi. Megjithatë, llogaritja e energjisë së komunikimit të të dhënave është komplekse, duke e bërë atë të vështirë për zhvilluesit që të profilizojnë aplikimet nga një perspektivë e konsumit të energjisë dhe optimizimit të modelit të trafikut.

Në këtë punim, ne jemi të interesuar dhe ekzaminojmë trafikun në sfond apo të njohur si "background" për një përdorues smartph one Android duke simuluar trafikun e të dhënave në një drejtim të caktuar për një protokoll specifik. Gjithashtu testojmë dhe një aplikim shumë popullor i cili mund të punojë duke transmetuar dhe në "sfond". Kjo duke pasur si qëllim vëzhgimin e fushave dhe elementeve ndikues në reduktimin e energjisë dhe kjo në këndvështrimin e ndërfaqeve të radio aksesit të rrjetit 3G/4G kryesisht. Për këtë qëllim, një ambient testimi si dhe një sërë pajisjesh fizike matëse HW e SW në PC si dhe një grup aplikimesh SW Android e Nokia janë përdorur për të përcaktuar konsumimin e energjisë dhe parametra të nevojshëm të rrjetit. Teste manuale dhe automatike janë ekzekutuar në një pajisje smartphone / inteligjente Nokia e Android të lidhur në rrjetin 3G/LTE për të hedhur dritë mbi modelet e trafikut si dhe energjisë së konsumuar për ndërfaqe, protokolle e aplikime të ndryshme. Matjet e përdoruara mund të jenë direkte ose të ruajtura në disa dosje specifike për proçesime të mëtejshme. Ruajtja e të dhënave të sistemit është përdorur më pas si një burim informacioni për të atribuar trafikun e të dhënave te modulet / protokollet SW të sistemit operativ Android dhe proceseve specifike në kod për zgjidhje efektive. Një teknikë e thjeshtë e menaxhimit të formës së trafikut të të dhënave me anë të planifikimit dhe vonesës të transferimit të këtyre të dhënave në drejtimin *Uplink* në shtresën specifike të strukturës së OS Android është implementuar si një modul shtesë për të dhënë më shumë eficencë energjie për trafikun e të dhënave në sfond. Si rezultat, konsumimi i energjisë i ndërfaqes 3G/LTE gjatë një pune aplikimesh si Facebook, Gmail e BBC News apo me bazë protokolli UDP është reduktuar deri në disa apo dhjetra % (në këndvështrimin e baterisë). Reduktimi i dedikohet menaxhimit më efektiv të volumit të trafikut, volumit të gjendjes së memorieve të protokollit RLC si dhe disa kohëve limit të transferimit të të dhënave në kanale 3G/LTE me shpejtësi të ndryshme secili.

Megjithëse qëllimi ynë është për një reduktim të konsumit të energjisë prej aplikimeve që punojnë apo transmetojnë të dhëna në sfond, kjo zgjidhje duhet të shtrihet dhe për trafikun interaktiv e protokolle të tjera që aplikimet apo rrjetat përdorin. Kjo i mbetet të ardhmes për shkak të kompleksitetit të ndërveprimit midis sistemit operativ, aplikimeve, protokolleve dhe parametrave të rrjetit të cilat mund të ndryshojnë dinamikisht pa njohjen nga ana e përdoruesit smartphonë (SW) apo aplikimeve që përdorin.

Abstract

Mobile devices in mobile communications have evolved rapidly in recent years and their use and performance are pushing modern battery technologies to their limits. The change in battery "sources" has come as a result of rapid changes in HW, SW technologies and Applications in today's smartphones. Energy efficiency investigations show that radio access networks are one of the main contributors to the total energy consumption seen on the smartphones. But with the complexity of today's HW and SW structure of smartphones, makes it necessary to identify and calculate consumptions in every HW,SW component or communication type. However, computing the energy of data communication is complex, making it difficult for developers to profile applications from a power consumption and traffic model optimization perspective.

In this paper, we are interested and examine the non-real time data traffic or known as the "background" for an Android smartphone user, by simulating data traffic in a particular direction for a specific protocol. We are also testing some very popular applications that can work in the background. This is aimed at observing the areas and elements of energy reduction and in the view of the 3G/4G radio access interfaces mainly. To this end, a test environment as well as a set of HW SW metering devices on a PC, and as well as a set of Nokia and Android SW applications have been used to determine the power consumption and necessary network parameters. Manual and automatic tests have been run on a Nokia / Android smartphone connected to the 3G / LTE networks to shed light on traffic patterns as well as power consumption for various interfaces, protocols and applications. The measurements used may be direct or stored in specific folders for further processing. System data storage was then used as a source of information to attribute data traffic to Android operating system SW modules / protocols and code-specific processes for effective solutions. A simple data traffic form management technique by planning and delaying this data transfer in the Uplink direction on the specific layer of the smartphone OS structure is implemented as an additional sw patch /module to give more energy efficiency for background data traffic. As a result, the power consumption of the 3G / LTE interface during applications such as Facebook, Gmail and BBC News or UDP based protocols can be reduced by some or tens of % (in battery levels mean). The reduction is due to more effective traffic volume management, RLC protocol memory capacity, and some time limit data transfer across 3G / LTE channels at different speeds each.

Although our goal is to reduce power consumption from applications that run or transmit data in the background, this solution should extend to interactive traffic and other kind of protocols that aplication or netwroks uses. This is the future because of the complexity of the interaction between the operating system, applications, protocols and network parameters which can change dynamically without the recognition of the smartphone user (SW) or the applications they use.

KAPITULLI I

1. HYRJE

1.1 Komunikimi dhe Njerëzimi

Komunikimi është pjesë përbërëse e njerëzimit. Nëpërmjet komunikimit, individët janë më mirë të pozicionuar për të kuptuar vetveten. *Sipas ekspertëve të teknologjisë, evoluimi i njerëzimit është i lidhur drejtpërdrejt me evoluimin e teknologjisë*. Kjo nënkupton se teknologjia po ndryshon me shpejtësi për tu përputhur me nevojat e njerëzimit. Në shoqërinë bashkëkohore, evolucioni i teknologjisë ka lehtësuar komunikimin në të gjitha nivelet e jetës. Për më tepër, teknologjia është duke ndikuar në mënyrën e komunikimit ndërmjet individëve. Nga sa më sipër është arritur një nivel më i lartë nëpërmjet futjes së smartphonave. Kryesisht në 10 vitet e fundit, bota u njoh me avancimin më të madh teknologjik të racës njerëzorë, *Smartphonët* ose *"Telefonat e Mençur"*, ndër ta iPhone, Samsung e Nokia. Për më tepër, teknologjia smartphone u siguroi internet njerëzve me akses të kufizuar interneti në zona të ndryshme. Teknologjia smartphone fuqizoi botën me aftësinë për të kryer kalimin nga aksesimi i televizionit në aftësinë për të parë drejtpërdrejt videot në telefoninë e mençur/smart për teknologjinë informative dhe shikimin e lajmeve kryesore nga e gjithë bota. Duhet thënë se teknologjia smartphone është një opsion i kostos efektive për shumë familje me të ardhura të ulëta në aksesimin e plotë të internetit. Smartphonët gjenden në shumë modele dhe formate të ndryshme të cilat ofrojnë një software të lehtë për tu përdorur me sistem operativ Android, iOS apo Windows mobile.

1.2 Motivimi

Ky punim është motivuar nga ideja e përdorimit masiv të telefonave pa tel "të mençur" apo "smart" në rrjetat *pa tel*, aplikimeve dhe shërbimeve që ato ofrojnë krahasuar me pajisjet e tjera duke lehtësuar aktivitetet e jetës së përditshme. Megjithatë, këto zhvillime në fushën e pajisjeve të mençura mobile shoqërohen me një sërë problematikash të shfaqura paralelisht me zhvillimin e tyre ku kërkohet një vëmendje më e madhe dhe kryesisht:

Për përdoruesin e pajisjes së mençur /smartphone jo rrallë hasen:

- Problematika të aplikimeve jo profesionale: si bllokime të pajisjes, sigurisë së informacionit, konsumit të lartë të të dhënave;
- Problematika të shterimit/konsumit të shpejtë të energjisë së përdoruesit, edhe në kushtet kur përdoruesi nuk e përdor shumë baterinë apo kur përdoruesi është në gjendje joaktive.

Ekzistojnë literatura dhe studime të shumta mbi këtë fushë pasi telefonia "*pa tel"*është bërë pjesë e përditshme e gjithësecilit dhe performanca e saj është një faktor që duhet konsideruar në vazhdimësi për një shërbim cilësor sa më të lartë. Shumë propozime, algoritme, simulime dhe zhvillime të reja në softwaret e rinj (OS) të pajisjeve smart (të mençur) vazhdojnë të studiohen edhe sot. Ndër to mund të rendisim:

- Shumëllojshmërinë e funksionaliteteve e cila rrit presionin mbi jetëgjatësinë e baterive, sinjalizimin në rrjet apo burime të tjera të brendshme e të jashtme;
- Pajisjet UE inteligjente / smartphone të përdoruesit kërkojnë gjithmonë e më shumë energji nga bateritë për funksionimin e tyre;
- Aplikimet e rrjetit të cilët janë konsumues i madh i fuqisë dhe në mënyrë të konsiderueshme reduktojnë kohëzgjatjen e baterisë;
- Bateritë të cilët janë një përbërës shumë i rëndësishëm i smartphonëve të sotëm dhe "autonomia" e së cilëve varet direkt nga ata;
- Ndryshimet domethënëse varen nga *Prodhuesi*, sistemi operativ mobile apo *OS* si dhe faktorë të tjerë si mbulimi i rrjetit, parametrat radio, tipet e aplikimeve, periodiciteti i transmetimeve etj.
- Teknologjia e baterive të smartphonëve nuk ka ecur përpara me të njëjtin vrull sikurse pjesa tjetër HW dhe SW e tyre, megjithatë kërkime po zhvillohen ne procese kimike qe te mundësojnë më shumë energji në fraksion të kohës;

Për të gjitha këto, reduktimi i shpejtë i konsumit të fuqisë së UE duket të jetë ndër fushat më "kritike" për tu zgjidhur, e cila ka zgjuar dhe interesin tonë për këtë fushë.

1.3 Qëllimi dhe kontributi

Të gjitha këto shërbime të reja në pajisjet Smartphone kanë provuar që janë konsumues të lartë të energjisë, në mënyra të ndryshme dhe shtrojnë kërkesa e nevoja të larta për telefonat smart, sikurse jetëgjatësia e baterisë dhe mundësitë e operimit me shpejtësi të lartë të të dhënave.

- Pyetja që shtrohet është se:
  - Sa energji e konsumuar nga lloji i aktivitetetit në rrjet nëpërmjet ndërfaqeve GSM/GPRS, 3G, Lte, WiFi etj në telefonat celulare të krahasueshme me njëri tjetrin?
  - Si mund ta reduktojmë energjinë e konsumuar nga aplikimet më të përdorshme duke përdorur secilën prej këtyre teknologjive?

Për t'i dhënë përgjigje këtyre pyetjeve, i jemi drejtuar një studimi të detajuar matjesh në kapitujt 5 e 7 për të vlerësuar energjinë e konsumuar nga transferimi i të dhënave përmes secilit rrjet apo ndërfaqe duke arritur në përfundimin se energjia e konsumuar është e lidhur ngushtë me karakteristikat e ngarkesës së të dhënave. Ajo nuk është e lidhur vetëm me madhësinë totale të transferimit por edhe me ndërfaqen apo rrjetin e përdorur gjithashtu edhe si RSSI, UL apo DL, transferime paralele etj. për të cilat do të flasim më gjatë në kapitullin 5.

Faktorët që më së shumti kontribuojnë në një cilësi të dobët përfshijnë: *kohëzgjatjen e lidhjes, mos-aksesimin në shërbime, dështime, ngadalësime, jetëgjatësi e shkurtër e baterisë* si edhe çdo gjë tjetër e komplikuar dhe konfuze për përdoruesin apo UE. Për këtë arsye, detyra e shumë prej nesh është të kuptojmë sesi *pajisja smart* (UE), *bateria*, *rrjeti* dhe *aplikimet* ndërveprojnë së bashku për të parë, përcaktuar dhe optimizuar me performancën e duhur secilin prej këtyre faktorëve në mënyrë që përdoruesi fundor dhe rrjeti të mund të optimizohen në mënyrë sa më eficente. Në interesin tonë është për faktorin "*jetëgjatësi e shkurtër e baterisë* ".

Një nga aspektet energjitike të komunikimeve ajrore është përdorimi me eficencë i ndërfaqes ajrore (apo i radio burimeve). Në raporte të ndryshme, për të arritur një eficencë të lartë të energjisë, investigohet implementimi i kryer në shtresën fizike dhe shtresën e rrjetit por edhe në nivele të tjera, për një optimizim sa më të mirë. Një llogaritje e duhur e infrastrukturës së energjisë do t'i ndihmojë të dy si zhvilluesit OS e të Aplikimeve, ashtu edhe prodhuesit e Smartphonëve.

Në këtë punim, do të:

- Identifikojmë dhe diskutojmë "fushat / problematikat" ku ekzistojnë problematika bazë të konsumit të energjisë dhe përmirësimet e mëtejshme ku duhet të fokusohemi për menaxhimin e fuqisë tek telefonat "e mençur/ smart".
- Masim dhe diskutojmë rëndësinë e konsumit të energjisë përmes disa komponentëve fizikë (HW) dhe aplikimeve SW. Prezantojmë një analizë të konsumit të energjisë për një model Smartphone. Këto rezultate mund të përdoren për krahasime mes smartphonëve të ngjashëm si HW apo për OS të ndryshëm.
- Evidentojmë vlerat e konsumit për kanal komunikimi si dhe parametrat e kohëzuesve në komunikimin 3G/4G të nevojshme për hapat në vazhdim.
- Diskutojmë modelin matematikor të energjisë së UE për të dhënat si dhe algoritmet për optimizimin e planifikimin e transferimeve të të dhënave për përmirësimin e konsumit të energjisë.
- Merremi me ndërtimin e një moduli SW në shtresën e mesme të një OS mobile (middleware) duke u përpjekur të ruajmë një QoS të pranueshëm për aplikimet "background" apo sfondi kryesisht. Zgjidhja është bërë për sistemin operativ Android si një sistem i hapur për zhvilluesit dhe vendorët smartphonë të cilët e personalizojnë këtë sistem sipas produktit përkatës.
- Kryejmë testime dhe rivlerësime të energjisë së konsumuar me dhe pa modulin SW në pajisjet smart UE si dhe do të evidentojmë detyrat që na dalin për të ardhmen.

Pra, para së gjithash detyra para nesh është të kuptojmë se si pajisja smart (UE), batteria, Rrjeti dhe aplikimet ndërveprojnë së bashku për të parë, përcaktuar dhe optimizuar performancën e duhur për secilin në mënyrë që përdoruesi fundor dhe eficenca e rrjetit të mund të optimizohen në mënyrë sa më produktive. Ne besojmë se një kombinim dhe koordinim më i mirë i parametrave të rrjetit dhe zgjidhjeve SW / OS të ruajtjes së energjisë (zgjidhje të maturuara) do të sjellin një përmirësim në jetëgjatësinë e baterisë së telefonave (pavarësisht zhvillimeve HW në fushën e inxhinierisë kimike).

1.4 Struktura e punimit

Punimi përmban tetë kapituj, të organizuar si më poshtë:

Kapitulli 1 *përshkruan në terma të përgjithshëm disa nga elementët kryesorë të dizertacionit. Bazat shkencore mbi të cilat bazohet ky studim përmenden shkurt në terma të thjeshtuar. Jepet një tablo e përgjithshme e nevojës për komunikim, qëllimi, objektivat për realizimin e punimit, metodika dhe metodologjia e përdorur me të cilat lidhet puna konkrete e zhvilluar.*

Kapitulli 2 *i dedikohet studimit mbi smartphonët apo telefonat e "mençur", zhvillimeve dhe aftësive apo mundësive të tyre për ofrimin e shërbimeve të ndryshme tek përdoruesit. Analizohet ndikimi që ato kanë në jetën e përditshme të gjithësecilit prej nesh, në kohën e sotme si dhe larminë e aplikacioneve që ato ofrojnë. Si edhe theksohet nevoja në rritje për kërkesa energjitike dhe problematikat e evidentuara në lidhje me konsumin e shpejtë të energjisë në këndvështrimin fizik dhe logjik.*

Kapitulli 3 *përmbledh të gjitha studimet e kryera nga studiues të ndryshëm në lidhje me problematikën e konsumit të energjisë në smartphonët e sotëm. Shumë prej tyre kryejnë matje, testime e simulime si dhe ofrojnë zgjidhje e propozime diskrete për përmirësimin energjitik nëpërmjet implementimit të algoritmeve/metodave specifike për zgjidhje eficente të transferimit të të dhënave si dhe për rrjedhojë reduktimin e konsumit të energjisë.*

Kapitulli 4 *diskuton mbi Rrjetat Mobile 3G e 4G, elementët e rrjetit për pjesën radio në interesin tonë si dhe faktorët ndikues në konsumimin e energjisë. Karakteristikë është ndarja e klasave të shërbimeve të cilat paraqesin kërkesa dhe sjellje të ndryshme të përdoruesve në rrjet. Analizohen në detaj ndërfaqet Uu e IU të rrjetit së bashku me protokollet specifike. Evidentohen protokollet përgjegjës për menaxhimin dhe transferimin e të dhënave si ai i RRC- i kontrollit të radio burimeve dhe RLC-i kontrollit të radio linkut (për memorizimin e të dhënave para transmetimeve). Përkufizohen 4 gjendje të pajisjes smartphone dhe kushtet apo limitet në volume trafiku apo kohë për ndryshimet midis këtyre gjendjeve. Gjithashtu, analizohen mekanizmat e komunikimit në rrjet që i mbajnë pajisjet smartphone të lidhur si dhe karakteristikat e trafikut në këto raste.*

Kapitulli 5 *studion më në detaj gjendjet e RRCs dhe implikimet apo problematikat e shfaqura në UE. Analizohen gjendjet e RRC dhe konsumi i energjisë për çdo gjendje si dhe kohëzuesit e pasivitetit dhe implikimet e tyre në konsumin e energjisë. Vëmë në dukje nëpërmjet matjeve specifike konsumin e energjisë për gjendjet e RRC si dhe për nxjerrjen e kufijve të memorjeve të RLC. Evidentojmë dhe shfaqim "gropën energjitike" si dhe problematikat kryesore të smartphonëve të sotëm. Japim shembuj konkretë të llogaritjes së kapacitetit të baterive si dhe tregojmë metodologjitë e matjeve për nxjerrjen e parametrave të nevojshme për studimin tonë. Kryejmë dhe një seri matjesh për një model Smartphone dhe evidentojmë fushat ku konsumi i energjisë është më i lartë si të dhëna për tu përdorur më pas për krahasime të mëtejshme. Në fund të kapitullit përmbledhim të gjithë problematikat e konsumit të energjisë në lidhje me transferimin e të dhënave si dhe disa teknika (metodologji) për planifikimin e këtyre të dhënave për një optimizim të gjendjes energjitike.*

Kapitulli 6*, bazuar në analizën e kapitujve të mëparshëm, propozon disa zgjidhje ose rrugë për përmirësimin energjitik dhe këto të kombinuara në grup ku pjesë e tyre është dhe një rrugë apo algoritëm efiçent për planifikimin e transferimit të të dhënave në drejtimin UL. Të dhënat do të transferohen në kushte apo gjendje të ndryshme energjitike nëpërmjet planifikimit të tyre në varësi të volumit të trafikut si dhe gjendjes së memorieve. Më tej, shpjegojmë logjikën e kësaj metodologjie të transfermit të të dhënave me anë të përdorimit me efikasitet të memorieve të RLC për një planifikim sa më efiçent të energjisë së paketave.*

Kapitulli 7 *trajton sistemin operativ mobile Androd, arkitekturën, veçoritë e bazuara në sistemin Kernel Linux si dhe makinën virtuale Dalvavik etj. Analizojmë në detaje "stakun" e sistemit operativ Android si dhe mundësitë e implementimit të zgjidhjes së treguar si një modul specifik i shtresës së kernelit apo si një patch Netfilter. Hapësira e përdoruesit do të ndërveprojë me shtresat më të ulta të OS dhe për rrjedhojë me modulin e propozuar me anë të thirrjeve të sistemit. Për këtë, mund të përdoret një aplikim libnetfilter i gatshëm i cili në ndërthurje me modulin e propozuar do të ofrojë një planifikim më efikas energjie të paketave. Netfilter është një kornizë e brenda kernelit Linux që jep mundësinë të interceptojë paketat dhe t'i modifikojë ato dhe në këtë mënyrë fluksi i paketave mund të kontrollohet, modifikohet në përputhje me planifikimin e transmetimeve të të dhënave për tip kanali e volumi. Tregojmë përdorimin me efikasitet të grepave netfilter si dhe implementimin e modulit. Gjithashtu, tregojmë kujdes në parametrat e nevojshëm të OS për shtimin e modulit të ri brenda stakut të sistemit operativ Android. Në fund, merremi me një seri testimesh me dhe pa modulin shtesë në OS të pajisjes mobile Android duke nxjerrë në pah përfitimin energjitik. Gjithashtu, evidentojmë problematikat në vijimësi dhe zgjidhjet për të ardhmen.*

Kapitulli 8 *jep përfundimet e nxjerra së bashku me problematikat e hasura nga të gjitha testimet e kryera si dhe gjithashtu vihet në dukje dhe puna jonë për të ardhmen.*

KAPITULLI II

2 "TELEFONAT E MENÇUR"(SMARTPHONËT) DHE PROBLEMATIKAT

2.1 Hyrje

Vitet e fundit, për më tepër gjatë dekadës së kaluar dhe kësaj dekade, telefonia *pa tel* ka pësuar një zhvillim të vrullshëm, kryesisht në drejtim të shërbimeve dhe pajisjeve të reja për përdoruesin. Nga një telefon i thjeshtë për komunikim zanor, telefoni celular u kthye në një pajisje multimediale me shumë shërbime të reja tërheqëse për përdoruesit dhe kryesisht shërbimet e të dhënave paketë (smartphone). Të gjitha këto u mundësuan nga evoluimi i rrjetave pa tel apo siç njihen ndryshe gjeneratat 2G (GSM/EDGE & 2.75G), 3G (UMTS/HSPA$^+$ 3.5G) dhe 4G$^+$ (LTE-A).

"*Një smartphone është një mjet i avancuar komunikimi me aftësi të përparuara telefonike e procesuese dmth. me aftësi të ngjashme kompjuterike".*

Pajisjet impresionuese ose smartphonët apo edhe "tabletët", janë produktet elektronike tejet të shitura sot për sot. Këto pajisje lejojnë aplikime të ndryshme si "vídeo streaming" apo lidhje të përhershme në rrjetet sociale nëpërmjet internetit mobile. Trafiku më i madh i Internetit "*pa tel*" përdoret për *web browsing* dhe pjesa e mbetur përdoret për mediat sociale dhe transferimin e dhënave apo dosjeve, lojërave në rrjet etj. Interneti *pa tel (wireless)* është gjerësisht i zhvilluar dhe shpejtësitë e të dhënave në përdorim po rriten ndjeshëm (disafish nga viti në vit). Smartphonët janë të pasjisur me shumë e shumë funksione HW. Shërbimet *pa tel* të media "streaming" po përdoren gjerësisht dhe volumi i trafikut ka filluar të okupohet kryesisht edhe nga shërbimet video. Komunikimet e çastit (instant) me zë, video e rrjetet sociale (SN) janë nga më të preferuarat, dhe kjo sjell një rritje të lartë të përdorimit të rrjetit të aksesit radio BSS, CN si dhe mjediseve e komunikimit midis tyre.
Pritet, për mos të thënë, që trafiku "wireless" (pa tel) dhe pajisjet mobile do të tejkalojnë trafikun fiks të të dhënave prej 2016 [14]. Cisco parashikon një 13-fishim të rritjes në trafikun mobile për 4 vitet në vijim, me një rritje 3 herë më shumë, krahasuar me trafikun IP fiks.

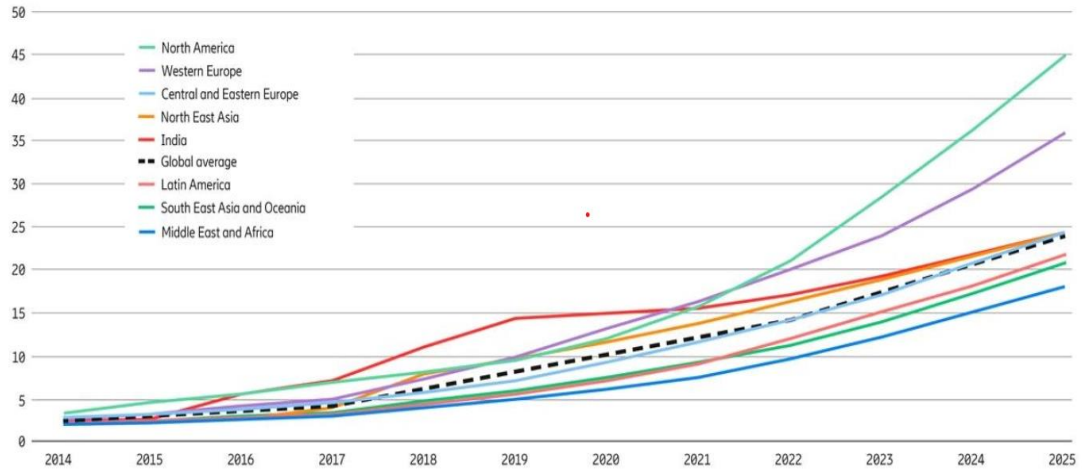


Figura 2.1: Parashikimi i përdorimit të të dhënave mobile

Të gjitha këto zhvillime i dedikohen funksionaliteteve në rritje që lidhen dhe me aplikime të reja e specifike që përdoren në Smartphonë dhe që ofrojnë shërbime nga më të shumëllojshmet e më të kërkuarat nga përdoruesi. Megjithatë, siç e përmendëm, një aspekt i rëndësishëm për kohën është edhe konsumi i shpejtë i energjisë. *Smartphonët e sotëm modern kanë disa ndërfaqe ajrore të ndërtuara për komunikim, kamera me rezolucion të lartë, pajisje GPS, të cilat konsumojnë mjaft energji etj*. Përdorimi i të gjitha këtyre shërbimeve shkakton rënie të shpejtë të energjisë së baterisë dhe mbinxehje të saj duke reduktuar ndjeshëm kohën operacionale të punës së pajisjes *pa tel* (UE). Në smartphonët modern, ndër tre "burimet" kryesore të konsumit më të shpejtë të fuqisë janë:

- CPU,
- Ekrani,
- Ndërfaqet e rrjetit.

Ne kemi parë se komponentët e ndërfaqes së rrjetit dhe ekranit dominojnë më së shumti konsumin e fuqisë. Për shembull, në Smartphone-in HTC magic Android (figura 2.2), me të gjithë komponentët në punë në kulmin e tyre, ekrani LCD dhe ndërfaqja 3G e rrjetit mobile konsumojnë respektivisht 45-50% dhe 35-40% të fuqisë së sistemit. Pjesa e mbetur e fuqisë konsumohet nga CPU't dhe nënsistemet e memorjes. Ajo mund të ndryshojë në varësi të modelit Smartphone (HW) dhe sistemit operativ (OS).

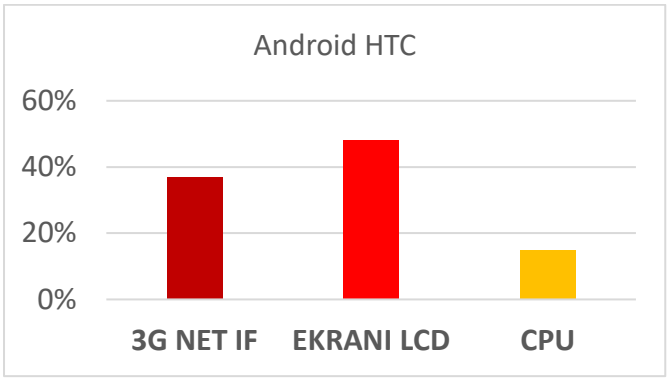


Figura 2.2: Shpenzimet e fuqisë për elementët në Android HTC

Një nga aspektet e komunikimeve ajrore (pa tel) është përdorimi me eficencë i ndërfaqes ajrore (apo i radioburimeve). Në raporte të ndryshme, për të arritur një eficencë të lartë, investigohet implementimi i kryer në *shtresën fizike* dhe *shtresën e rrjetit* por edhe në nivele të tjera si atë aplikative, për një optimizim sa më të mirë si dhe eficencë energjitike. Por jo gjithmonë kjo mund të arrihet pasi ka dhe faktorë të tjerë që ndikojnë si psh. tipet e aplikimeve, lloji i shërbimit dhe ndërveprimet me rrjetin.

Një nga komponentet më të dukshëm të konsumit të lartë të fuqisë në smartphone është ndërfaqja e rrjetit celular 2G, 3G e 4G. Smartphonët zakonisht punojnë në aplikime bazë "Cloud" dhe shërbime që kërkojnë akses të vazhdueshëm në internet për të mbajtur të sinkronizuar UE me informacionet më të fundit. Kështu që aktiviteti i të dhënave, i aplikimeve smartphone, i shkakton ndërfaqes wireless të rrjetit të aksesit p.sh 3G, të oshilojë midis gjendjeve aktive (konsumues i lartë i energjisë) dhe idle. Aplikimet me bazë "Cloud" në smartphone ekzekutojnë aktivitetet e të

dhënave në mënyrë të pavarur nga njëri-tjetri, dhe kështu një numër asinkron i aktivitetit të të dhënave rritet me numrin e aplikimeve bazë "Cloud" të instaluara dhe aktive/në punë. Aplikimet mund të bëjnë reklamime (p.sh banners) ose shërbime sfond (background) për të dërguar të dhëna përdoruesi specifike te aplikime të tjera ose website, të cilat paralelisht rrisin trafikun e të dhënave të aplikimeve të rrjetit në UE. Por pa ndonjë dëshirë apo ndërveprim të përdoruesit shumë aktivitete të dhënash në sfond (background) të smartphonëve ndodhin në vazhdim. Kjo në sajë të llojit të komunikimit dhe mënyrës së programimit të aplikimeve.

2.2 Telefonat e mençur (smart) dhe ndikimi i tyre në jetën tonë

2.2.1 Ndikimi dhe sfidat për përdoruesit e Smartphonëve

Smartphonët mund të kryejnë ndër detyrat më të nevojshme që ju bëni me kompjuterin tuaj. Ai mund të lidhet me internetin, emailet, të përdorë faqet e rrjeteve sociale dhe të organizojë jetën tonë. Në këtë mënyrë ato po kthehen në zëvendësues të shpejtë të kompjuterave të cilët ishin pjesa më e pazëvendësueshme e teknologjisë në jetët tona. Po ashtu dhe për përdoruesit smartphone ekzistojnë me mijëra aplikacione ku shumë prej tyre janë dhe shkaku kryesor i sjelljes së "keqe" të pajisjeve smartphone, duke filluar që nga harxhimi me shpejtësi i baterisë, çështje të lidhura me sigurinë gjatë kohës që jemi në internet, aplikacione ose përgjigje të ndërvarura të UE dhe kështu me radhë [9, 40, 114].

Zbërthimi i aplikacioneve të mira dhe zgjidhjes proaktive nga rrjetet celulare nuk është një detyrë e lehtë pasi ato punojnë të varura nga njëri-tjetri. Aplikacionet *gjithmonë në linjë / "always online"* si mesazhet e çastit (IM) dhe shërbimet e rrjeteve sociale (SNS) etj. janë të njohura në mesin e përdoruesve celularë. *Këto aplikacione jo vetëm zënë burimet e lidhura me mbartësin e rrjetit pa tel, PDP, dhe IP, por gjithashtu i kërkojnë pajisjeve të klientit të dërgojnë mesazhe heartbeat (mbijetese) në server çdo disa herë në minutë në mënyrë që të qëndrojë në linjë.* Kjo gjithashtu krijon një mbingarkesë të rënduar sinjalizimi në rrjet. Shumë smartphonë (apo telefona "të mençur") i bashkohen automatikisht rrjetit dhe aktivizojnë përmbajtjet PDP menjëherë pasi lidhen në rrjet, në mënyrë që të aksesojnë internetin dhe të sigurojnë përditësim të informacionit kudo dhe kurdo (kjo është në varësi të konfigurimit të çdo terminali nga përdoruesi). Për të ruajtur energjinë, Smartphonët dhe Operatorët celularë shpesh adoptojnë teknologjinë e përgjumjes së shpejtë apo FD në UMTS e LTE. Nëse nuk ka transmetim të dhënash brenda një kohe të shkurtër (zakonisht 1 deri në 10 sekonda), smartphonët automatikisht ndërpresin lidhjen e rrjetit "pa tel" dhe kalojnë në gjendjen e *Qetë* (Idle). Lidhja do të rivendoset kur transmetimi i të dhënave të jetë i nevojshëm. Disa kategori të internetit aktual të lëvizshëm, aplikimeve dhe karakteristikat e tyre kryesore jepen si në tabelën 2.1.

Kategoria	Përshkrim	Aplikacione Tipike	Karakteristika
Voip	Thirrje audio dhe video	Viber, Skype, Tango, Face time	paketa të vogla, të vazhdueshme
Streaming	Streaming media si video HTTP, audio dhe P2P	YouTube, Pandora, PPStream	paketa të mëdha, të vazhdueshme

Web browsing	Faqe web në internet	Web browsers tipikë, Safari, Opera, Bing	paketa të mëdha, jo të shpeshta
SNS	Faqe të rrjeteve sociale	Facebook, Twitter, Instagram	paketa të mëdha, jo të shpeshta
Email	Mail-e përfshirë Webmail, POP3, SMTP	Gmail, Yahoo, Hotmail, Outlook	paketa të mëdha, jo të shpeshta
Lojrat	Lojrat në celular	Cars, AngryBirds	paketa të mëdha, të vazhdueshme

Tabela 2.1: Disa kategori të internetit aktual (të lëvizshëm) dhe karakteristikat e tyre kryesore

2.2.2 Aftësitë e terminaleve celulare inteligjente

Me zhvillimin e internetit celular, aftësitë e rrjetit dhe aftësitë e Smartphonëve po ndryshojnë shpejt. Në ditët e sotme, shumica e Smartphonëve përputhen me 3GPPRelease 6 dhe më tej. Gjithnjë e më shumë smartphonët mbështesin teknologjinë *HSPA+*karakteristika si dhe modulimet 64QAM, shumë hyrje-dalje (multi-input & multi-output - *MIMO*), lidhje pakete të vazhdueshme, dhe kanale *Cell_FACH* të zgjeruar. IPad-i i ri në përputhje me 3GPPRelease 7 ka një aftësi downlink të Cat. 14Mbit/s deri në 21.1Mbit/s. IPad-i ri mbështet karakteristikën DC-HSDPA si funksion në Release 8, me një aftësi Downlink/DL të Cat. 24Mbit/s deri në 42Mbit/s. Ndërkohë që modele të fundit suportojnë edhe shpejtësi të dhënash në teknologjinë 4G/LTE-A deri në 150/300 Mbps e më lart (ku mendohet se mund të shkojë dhe 1Gbps me përdorimin e shumë frekuencave apo antenave të njohur si MIMO).

Por gjithashtu madhësia e ekranit Smartphone, përbërja dhe rezolucioni janë përmirësuar me shpejtësi. Aftësitë e Smartphonëve për procesim, aftësia e ndërfaqeve multiradio si dhe ekranet e tyre janë bërë gjithnjë e më të mëdha. Rezolucioni më i lartë i ekranit të UE në mënyrë të drejtpërdrejtë çon në rritjen e volumit të trafikut të të dhënave në dërgim/Tx = UL e marrje/Rx = DL. Të gjitha këto punojnë të varur nga bateritë. Konsumi i shpejtë i energjisë është bërë një pengesë për Smartphonët dhe vazhdon të mbetet një sfidë edhe sot. Në figurën e mëposhtme janë paraqitur lloje të ndryshme celularësh Smartphone dhe OS-të e tyre. Sikurse shihet sistemi operativ Android mbetet më dominuesi për shkak të fleksibilitetit dhe shumëllojshmërisë së aplikimeve. Tendeca sot është ritur dhe më shumë me përdorim më në masë të smartphonëve Andorid.

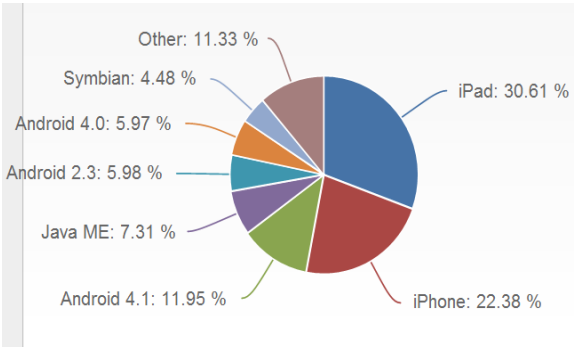


Figura 2.2: Aksionet e tregut sipas OS celulare/tablet deri më Jan 2014 [24]

Për shembull, rrjetet sociale dhe të pozicionimit të përdoruesit janë në një rritje tepër të lartë e shumë popullore sot për sot. *Rritja e bandwidth-it dhe lejimi i njëkohshëm në punë i disa aplikimeve në sfond/background ka rritur kërkesën për më shumë bite për transmetim ose e përkthyer ndryshe më shumë energji e kërkuar dhe se më parë.*

Analizat e tregut tregojnë se Smartphonët kanë tejkaluar numrin e kompjuterave në përdorim rreth fundit të vitit 2015 dhe përgjatë 2016. Ato do të jenë pajisja më e zakonshme dhe e përdorshme për të hyrë në internet. Çmimet dhe ofertat nga operatorët mobile i kanë bërë smartphonët të mundur edhe për shtresat me të ardhura të ulëta, çmimet e tyre sa vijnë e ulen kohë pas kohë.

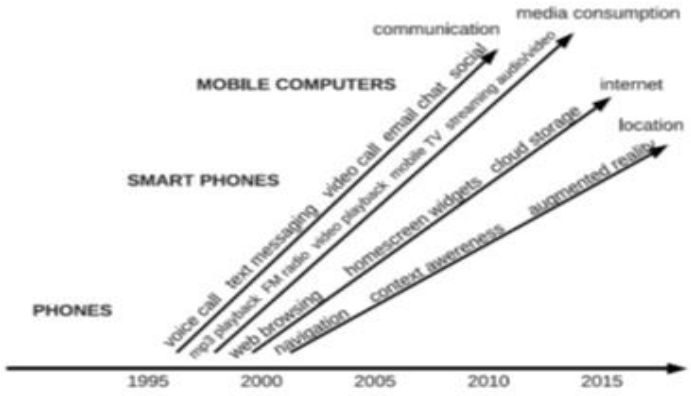


Figura 2.3: Evoluimi dhe rastet e përdorimit në pajisjet mobile [2]

2.2.3 Aplikimet, periodiciteti dhe problematikat

Në figurën 2.2 janë paraqitur aksione të tregut të Sistemeve Operuese Celular/Tablet deri më Janar 2014. Shihet se për çdo lloj sistemi software operimi *mobile* (OS) ka mijëra dhe qindra-mijë aplikacione të disponueshme dhe me siguri shumica prej tyre nuk përmbushin ndonjë cilësi të QoS [26] dhe ato mund të jenë shkaku kryesor i shumë çështjeve/problemeve për përdoruesit fundorë dhe rrjetet celulare. *Në varësi të kërkesave të tyre, këto aplikacione mund të shkaktojnë ngarkesa të larta (të memories, CPU-s) dhe ndërprerje të shërbimeve apo bllokim të pajisjes.* Për shembull, mediat sociale dhe aplikacionet email janë vazhdimisht duke kërkuar një lidhje në rrjet për të kontrolluar për përditësime.

Aplikacionet që kërkojnë komunikim të vazhdueshëm në rrjet mund të gjenerojnë ngarkesa të rënduara trafiku që rrjetet mund ta kenë të vështirë për ta ndaluar. Përdoruesit e sofistikuar nuk po shtojnë vetëm aplikacione të reja, ato po i "modifikojnë" aparatet, i shtojnë ose heqin funksionalitet dhe e rindezin sistemin operativ për të personalizuar dhe përmirësuar telefonat e tyre (rooting në Android). Kjo mund të shkaktojë një sërë çështjesh/problematikash shtesë. Këto çështje paraqesin një problem të vazhdueshëm dhe për rrjetat e operatorëve, pavarësisht përpjekjeve më të mira të prodhuesve të pajisjes *pa tel* të përdoruesit (UE), zhvilluesve dhe programuesve të aplikacioneve.

Aplikimet në smartphonë mund të ndahen në aplikime, në përdorim në sfond (background) dhe në përdorim në plan të parë aktive (ndryshe si foreground). Shpesh për aplikimet të cilat kryejnë detyra sporadike në sfond në rrjet, një grup kodesh apo dhe periodicitet (në nivel të ulët binar) drejton/çon në përdorimin e rrjetit në mënyrë "aktive" nga aplikimi. Këto tipe aplikimesh kanë tipare të ndryshme të aktivitetit në sfond të rrjetit, pasi disa sesione mund të jenë të gjata dhe

përfshijnë një trafik domethënës, ndërsa disa të tjera mund të jenë shumë të shkurtra dhe shumë periodike, sikurse aplikimet për mesazhet "mbaj-gjallë" që shkëmbehen çdo 10, 30 apo 60 sekonda për shembull. Në praktikë shumë aplikime në sfond mund të ndodhin në çdo moment kohe, dhe ndërfaqja radio mund të shkojë në gjendjen FD vetëm nëse asnjë nga aplikimet është duke komunikuar.

Kjo në mënyrë të natyrshme redukton energjinë. Për Qian at al.[3] kemi: ***a)*** transferimet periodike të cilat janë shumë prevalente në trafikun e smartphonëve të sotëm ku në 20% të sesioneve kemi ato më të gjata se 1 minutë brenda një transmetimi, ***b)*** transferimet periodike të cilat janë të vogla dhe të shkurtra me një sasi transferimi prej rreth 1.1 KB dhe në 90 % të rasteve janë rreth 7 sekonda të gjata, ***c)*** vlera e 60 sekondave dominon peridicitetin, i cili duket sikur është vendosur nga zhvilluesit në një kuptim më të gjerë.

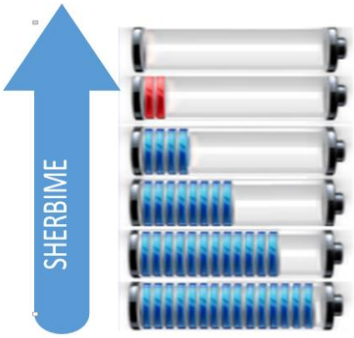


Figura 2.4 Marrëdhënia shërbime - konsum i energjisë në UE

Sistemet operative të smartphonëve të sotëm tentojnë të kontrollojnë më mirë punën në sfond të aplikimeve duke përcaktuar deri edhe profilet me përdorimet e tyre. Nëpërmjet kësaj ata mund të bllokojnë ose jo punën në sfond.

2.2.4 Përsëritja e kërkesave, sinjalizimi dhe energjia e harxhuar

Smartphonët duhet të jenë gjithmonë të lidhur në rrjet dhe ata përpiqen të mbajnë veten aktiv nëse ndodh ndonjë dështim. Kërkesa e përsëritur për aktivizim të ri sinjalizimi çon në një ngarkesë të panevojshme sinjalizimi në rrjet. Nëse ndodhin gabime të papritura në rrjet, kërkesat e përsëritura të aktivizimit mund të shkaktojnë një mbingarkesë të rëndë në rrjet (problemet në stacionet shërbyese apo *Node B*, avari në nyjet qëndrore si *Porta GW*, *Firewall* etj). Nga një sërë testimesh është parë se konsumi të dhënave i një Smartphoni është ~ 10% i trafikut *data* që një laptop gjeneron, por smartphonët me aplikimet e tyre i mbajnë të lidhur përdoruesit në *sfond (background)* në rrjet qindra deri në mijëra herë në ditë, duke shkaktuar trafik shumë më të lartë nga sa mund të imagjinohej nga planifikuesit e rrjetave mobile/*pa tel*.

Trafiku i sinjalizimit konsiston në mesazhe të vogla në sfond (background) që shkëmbehen midis pajisjes UE dhe rrjetit për të vendosur, mirëmbajtur dhe mbyllur një lidhje. Vendori Nokia [12] nëpërmjet testimit të një loje popullore në rrjet dhe matjes së të dhënave dhe trafikut të sinjalizimit të gjeneruar në dy raste, nga loja e luajtur në një laptop dhe nga loja e luajtur në një smartphon, në këtë rast në një iPhone me OS version 4.1 e më i vonshëm, u vu re se laptopi gjeneroi 188 mesazhe sinjalizimi ndërsa smartphoni gjeneroi **10 herë** më shumë, me rreth 1996 mesazhe gjatë 30 minutave aktivitet.

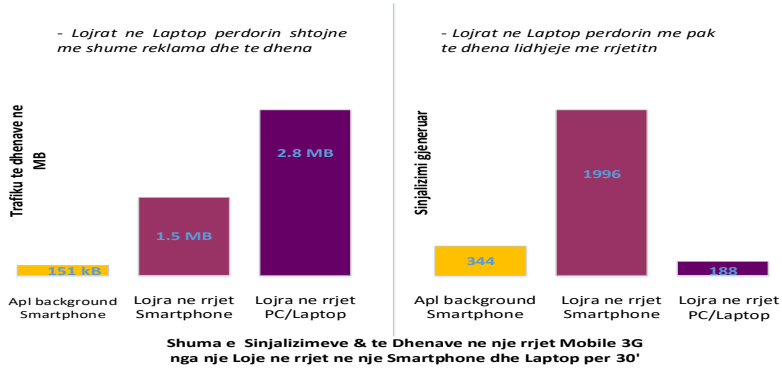


Figura 2.5: Testime në Lab të Nokia në lidhje me sinjalizimin e smartphone [12]

Mesazhet "heartbeat" (mbijetesës) për shumicën e aplikimeve mirëmbajnë lidhjet me serverat dhe përditësojnë statuset e tyre. Heartbeat-et e shpeshta së bashku me shërbimin e ri FD (përgjumje e shpejtë - fast dormancy) për smartphonët janë shkaku kryesor i sinjalizimeve masive në rrjetat *pa tel* sikur shihet dhe në figurën 2.6. Disa aplikime gjenerojnë një sasi të madhe të sinjalizimit e cila varet nga mënyra e programimit të aplikimit apo thënë ndryshe nga natyra e saj. *Sinjalizimi i shpeshtë sjell një ngarkesë të lartë (në procesorët/CPU-t e elementëve) në rrjetin radio dhe pjesën qendrore (bërthamë) të rrjetit, por gjithashtu mban në gjendje pune dhe ndërfaqen e rrjetit të pajisjes së përdoruesit duke* ***reduktuar ndjeshëm energjinë e tij***.

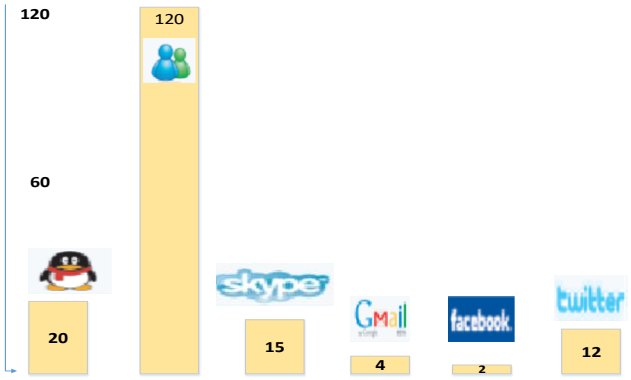


Figura 2.6: Ngarkesa e sinjalizimit në një rrjet për tipe të ndryshme aplikimesh (Android) [12]

2.3 Balanca që kërkohet

Për të përmirësuar perceptimin e përdoruesve për pajisjet mobile smartphonë, zhvilluesit a aplikimeve përpiqen të zgjidhin konsumin e shpejtë të energjisë të aplikacioneve të tyre. Megjithatë, zakonisht kjo nuk bëhet në mënyrë të përshtatshme dhe aplikacionet ekzistuese vuajnë përsëri nga defektet e shkaktimit të konsumit të shpejtë të energjisë. Sipas një studimi të bërë nga autorët siP. Vekris:" 55% e 328 aplikacioneve që përdorin "wakelocks" nuk ndjekin politikat për "no-sleep bugs "[2012]. Disa aplikacione të mëdha janë lëshuar me probleme të mospërgjumjes / "No-Sleep"[1].

Por për shumë vendorë pajisjesh Smartphone është i rëndësishëm për gjetjen e një balance në mënyrë që ekselenca e një aplikimi të mos shkaktojë kosto ekstra psh. për harxhimin e shpejtë të baterisë së UE apo rritjen masive të sinjalizimit në rrjet. Rrugët që një zhvillues aplikimesh dhe vendorë pajisjesh smart mund të ndihmojnë është që të gjendet një balancë optimale midis performancës së përdoruesit, jetëgjatësisë së baterisë dhe volumit të rritjes së sinjalizimit:

- *Optimizimi i transmetimit të të dhënave* për madhësinë dhe minimizimi i madhësisë së mesazheve "keep-alive" në mënyrë që të utilizohet përdorimi i Cell_FACH/RACH.
- *Optimizimi i përdorimit të update-ve prezente* dhe mesazheve të update-it të statusit marrin avantazh nga "caching" dhe opsioni i reduktimit të shkarkimeve të tilla si imazhe apo attachment (dokumenta të bashkëlidhur).
- *Shmangia e protokolleve që kërkojnë poll-in (votim). Mbyllje e lidhjes së të dhënave,* kur nuk ka më nevojë për lidhjen.
- *Zgjidhje e protokollit TCP mbi UDP,* derisa TCP kërkon më pak mesazhe "keep-alive" dhe kjo është e rëndësishme për aplikimet "gjithmonë aktive"si psh "*push email"*.
- *Zgjedhje periodike e transferimit të të dhënave* në burst sesa transferim sipas dëshirës (psh në DCH në 3G e LTE). Utilizimi i rrjetit të aksesit nëpërmjet APIs optimizuese (si psh aplikimet njoftuese) në mënyrë që të sinkronizojë transaksionet me rrjetin. Përdorimi i *mesazheve të buffering/agregimit* midis aplikimeve të ndryshme sa më shumë të jetë e mundur. *Reduktim i ndërveprimit me rrjetin* kur aplikimi nuk është në përdorim aktiv.

Kërkesa apo update të shpeshta me sasi të vogla të dhënash në kërkesë mund ta ngarkojnë rrjetin me sinjale ekstra dhe të harxhojnë jetën e baterisë të smartphonëve. Ndërfaqja radio e smartphonit duhet që herë pas herë të lëvizë nga gjendja e tij e *"fjetur/qet*ë" në gjendje të "*lidhur"*për çdo përpjekje lidhjeje. Për të ruajtur jetën e baterisë, ndërfaqet radio preferojnë të qëndrojnë në gjendje *të fjetur* sa më gjatë të jetë e mundur dhe të shkojnë në gjendje *të lidhur* në mënyrë sporadike për të transferuar sasi të mëdha të dhënash dhe më pas të rikthehen në gjendjen e përgjumur sa më shpejt që të jetë e mundur. Ka pasur përmirësime të vazhdueshme në standardet e rrjeteve celulare si psh. 3GPP ka përfshirë rregullatorët e kohës (si T_1 dhe T_2) që mund të ndihmojnë me sjelljet "llafazane" të aplikacioneve. Një rregullim më i mirë i tyre do t'i ndihmojë të dyja palët; si përdoruesin dhe rrjetin. Gjithashtu ka shumë studime që parashikojnë aktivitetet e përdoruesit bazuar në historinë e mëparshme dhe në këtë mënyrë përdoruesi mund të menaxhohet më mirë (vlerat RSSI, drejtimi, aplikacionet e përdorura dhe të tjera janë pjesë të këtyre studimeve).

Nga ana tjetër me futjen e smartphonit, zhvilluesit e aplikacioneve duhet të konsiderojnë përmirësime në aspektet e mëposhtme:

- Për dizenjimin standard, faktorë të tillë si paketat e vogla, bartësit e eficencës, arkitekturës së rrjetit dhe optimizimi i protokolleve të shtresave duhet të merren në konsideratë.
- Për rrjetet ekzistuese, modelet origjinale të trafikut kanë ndryshuar, ka ndryshime software e hardware, janë rikonfiguruar parametra rrjeti dhe karakteristika/shërbime të reja janë aktivizuar.
- Për smartphonët dhe aplikacionet, një situatë e favorshme pritet nga konsumi i burimeve të rrjetit dhe përvojës apo eksperiencës së përdoruesit.

Këto zgjidhje nuk mund "të zëvendësojnë" rikonstruksionet në rrjet ose zgjerimin e kapacitetit për të përmbushur kërkesat e përdoruesve gjithnjë e në rritje, si dhe sinjalizimit dhe trafikut të të dhënave. Operatorët e telefonisë celulare kryesisht në teknologjinë 3G e 4G duhet të konsiderojnë impaktet e smartphonëve dhe të marrin masat e duhura për të reduktuar impaktin e tyre nga sjellja "agresive". Operatorët mund të ndërmarrin veprime korrigjuese për të minimizuar impaktin në

kapacitetin dhe performancën e rrjetit. *Nga ana tjetër vetë përdoruesi duhet të marrë në konsideratë llojin e aplikacioneve të përdorura dhe impaktin e tyre në celularin e tyre (kjo në lidhje me performancën e UE, si dhe në lidhje me baterinë).* Një pikë e rëndësishme në rrjetat celulare është dhe reduktimi i shumës së të dhënave të transferuara pa kompromentuar *semantikën* e aplikimeve. Krahasuar me rrjetat WiFi, kjo çështje është më kritike veçanërisht për rrjetat mobile. Nga këndvështrimi i bartësve (frekuencave) rrjetat celulare operojnë nën ndrydhjen (limitin) e burimeve. Edhe një reduktim i vogël i trafikut total të volumit me 1% mund të çojë në ruajtje të shpenzimeve të mëdha për bartësit apo frekuencat. Gjithashtu, edhe një reduktim i konsumit të shpejtë të baterisë çon në një reduktim të karikimeve të shpeshta si dhe një ruajtje të konsumit global të energjisë. Të mirat janë gjithashtu të dukshme nga këndvështrimi i përdoruesve sepse sa më pak të dhëna të transferuara aq më pak shpenzime celulare, përmirësim të eksperiencës së përdoruesit dhe reduktim të konsumit të energjisë së pajisjes së UE ka.

Sikurse kemi përmendur në fillim, puna jonë konsiston në implementimin e një zgjidhjeje "middleware" për telefonat smart me OS bazë Android i cili do të ndihmojë në planifikimin e transferimit të të dhënave si dhe në uljen e harxhimit të energjisë duke marrë në konsideratë tipin e trafikut, volumet e trafikut, kohën e aktivitetit. Androidi është zgjedhur përmes shumë të tjerëve për shkak të rritjes së shpejtë të popullaritetit në smartphonët dhe si një SW shpresëdhënës, i cili është një OS i hapur për zhvillime të mëtejshme dhe për vendorë Smartphonësh HW të ndryshëm. *Edhe pse Android jep një grup aplikimesh me bazë kernel Linux-i dhe middle-ware për zhvilluesit e aplikimeve të telefonave, ai nuk utilizon/v*ë në përdorim *shumë shërbime standard të kernelit të Linux.* Kështu që kërkesa e optimizimeve për eficencë energjie është dhe mbetet kritike për kohën.

2.4 Problemet energjitike

Problemet energjitike ose të emërtuara si "ebugs" nga disa studiues në këtë fushë, përcaktohen si një problem/error në sistem, ose dhe në aplikim, OS, harware, firmware ose shkaqe të jashtme të paparashikuar duke shkaktuar një konsum të lartë energjie. Këto mund të vijnë si pasojë e gabimeve në programim, përdorim jo të rregullt të API, dizajn jo i rregullt i aplikimeve ose i specifikimeve OS (driverat e pajisjes smartphone/UE), ndërveprime komplekse midis komponentëve hardware të smartphonëve, ndryshimet e kushteve të jashtme (p.sh probleme në serverin lidhës, sinjal i dobët i valëve radio, etj). Eprobl nuk çon në një "dëmtim" të aplikimit. Aplikimi dhe i gjithë sistemi vazhdojnë të punojnë "normalisht", ndërkohë që konsumi i energjisë rritet në mënyrë të paparashikueshme duke sjellë një shkarkim shumë të shpejtë të baterisë.

- **E*probl* i pjesëve fizike (Hardware):**

-*Bateria:* mund të ndodhë si shkak i baterive të vjetra e prishura, psh. nuk mban të gjithë ngarkesën e karikimit dhe rezultojnë në shkarkime të brendshme dhe mbinxehje.

- *Dëmtime të jashtme të HW:* Dëmtime të butonave të jashtëm të telefonit duke shkaktuar "prekje" të rastit, si rrjedhoje vetë-aktivizime procesesh ose aplikimesh etj.

- *Kartat e memories SD*: Kartat e korruptuara SD mund të trigerojnë aplikime, të vendosen në gjendje "loop" / pambarim, ku ato në mënyrë të përsëritur tentojnë të aksesojnë HW.

- **E*probl* të software:**

- Problematika *OS:* updatet e OS, nga përdoruesi ose me forcë, rezultojnë në probleme të OS. Shkaku i problemeve janë proceset "llafazane" si psh. procesi "suspend" në Android i cili ekzekutohet në sfond (background) duke e mbajtur CPU-n zgjuar dhe duke shkaktuar shkarkim të shpejtë të baterisë dhe ndryshime në konfigurim si profile jo korrekte të set CPU të cilat rezultojnë në *overclocking* dhe *underclocking*.

- **E*probl* të aplikimeve:**

- Problematika "*No-Sleep"*: është një situatë ku një aplikim kërkon të zgjojë një mekanizëm lock (bllokim) për një kompenent i cili zgjon komponentin, por nuk e lëshon lock-un (bllokuesin) edhe pse puna është kompletuar. Problemet mund të vijnë si shkak i programimeve të gabuara, kondicionet që bëjnë bllokimin e lëshimit të lock.

- Problematika "*Loop"*: një problem "loop"(i mbyllur) ndodh kur një pjesë e një aplikimi hyn në një gjendje përsëritjeje duke kryer detyra periodike të panevojshme, duke shkarkuar ndjeshëm baterinë. Detyrat periodike mund të jenë update me polling të panevojshme të rrjetit, ose përdorimi i ndonjë kompenenti tjetër në loop. Nëse një server në distancë "bie", fjalëkalimi i emailit ndryshohet, klienti në mënyrë të përsëritur tenton të lidhet me serverin, ose performon autentikim emaili.

- Problematika "*Immortality"*: një defekt "i pavdekshëm" është rrjedhojë e një aplikimi llafazan që shkarkon baterinë, derisa të jetë "vrarë"(killed) nga përdoruesi, dhe rishfaqet duke hyrë në të njëjtën gjendje duke shkaktuar shkarikim të baterisë.

Të gjitha këto lloj problematikash të evidentuara krijojnë një kompleksitet në UE dhe në rrjet duke krijuar një "degjenerim" të shpejtë të baterisë dhe perceptimit të përdoruesit. ***Për të zhvilluar teknika efiçente energjie, pikë se pari duhet të kuptojmë se si energjia është konsumuar në një pajisje mobile.*** Një pajisje mobile konsiston në shumë komponentë hardware. Komponentët hardware të telefonit janë konsumuesit aktual të energjisë.

Për një komponent hardware, fuqia e tij e konsumuar konsiston në dy pjesë:

- *fuqi statike* dhe
- *fuqi dinamike* e konsumuar.

Konsumimi i fuqisë statike, e njohur si rrjedhja qëndron/vazhdon, pavarësisht gjendjes së komponentit hardware është ajo që e mban pajisjen hardware në gjendje pune/standby. Ajo varet nga karakteristikat fizike të komponentit hardware. Në kontrast me fuqinë dinamike të konsumit e shkaktuar nga aktiviteti i hardware-it, të cilët janë të kontrolluar nga software-i i cili ekzekutohet në krye të hardware-it. Ngarkesa e gjeneruar nga software-i, duke ju shtuar karakteristikave fizike, përcaktojnë shumën e fuqisë dinamike të konsumuar.

Në kapitullin në vijim shqyrtohen studimet e kryera në fushën e konsumit të energjisë në smartphonët e sotëm dhe zgjidhjet apo rrugët për zgjidhjen e këtij "problemi".

KAPITULLI III

3 STUDIME TË TJERA NË LIDHJE ME KËTË FUSHË

Ky punim studion konsumin e fuqisë duke u fokusuar veçanërisht në pajisjet mobile smartphonë dhe mobile OS. Konsumi i energjisë në smartphonet e sotëm është bërë një pikë e rëndësishme për studiuesit dhe prodhuesit smartphonë. Kujdesi për konsumin e energjisë të aplikimeve të ndryshëm dhe zhvillimi teknologjik në komunikimet wireless janë gjithmonë në rritje. Ky kapitull përmbledh studime të ndryshme në lidhje me fushën e analizimit, matjeve dhe optimizimit të energjisë së konsumuar në komunikimet wireless në pajisjet mobile si dhe teknikat e ndryshme për ruajtjen e energjisë. Për më shumë ne përmendim studimet e ndryshme të fokusuara në implementimin e ruajtjes së energjisë në smartphonët e sotëm duke përfshirë dhe fusha të tjera që na interesojnë për të ardhmen. Studimet e zhvilluara mbi energjinë e UE përfshijnë eksperimente për konsumin e energjisë në pajisjet mobile duke përdorur tools-e ose matës fizik e software (in-built) në pajisjet mobile ose pajisje të jashtme matëse. Gjithashtu, aty ekzistojnë tipe të ndryshëm të transmetimit të të dhënave, llojit të trafikut, dhe algoritmit të kontrollit të burimeve radio për reduktimin e konsumit të burimeve në rrjetat celulare), transferim i njëhershëm (piggyback), transferim me zgjedhje me vonesa (batching), algoritmet TailEnder & TailTheft, Timer Alignment [7, 29, 36,] dhe algoritmet për kontrollin e radio burimeve (si FD apo algorimi TOP) për reduktimin e konsumit të burimeve për transferimin e të dhënave celulare. Punët tona paraardhëse[67] përpunojnë këtë studim, duke kryer matje fizike, duke përdorur një UE si modem që izolon energjinë për transferimin e të dhënave.

- Në lidhje me matjet me mbulimin, ndërfaqet e rrjetave, CPU-n:

Një studim shumë i rëndësishëm duke përdorur SW NEP në telefonat e Nokia (OS Symbian) nga Balasubramanian et al. [7] raporton kategorizimin e tre tipeve të ndryshëm të energjisë në rrjetat celulare: ramp (e përpjetë-fillestare), e transferimit dhe e bishtit (tail). Ata studiojnë dhe karakterizojnë energjinë e konsumuar të aktivitetit të rrjetit në GSM/2G, 3G, 4G dhe WLAN, me qëllim reduktimin e energjisë së konsumuar në aplikimet mobile që përdorin këto teknologji rrjeti. Eshtë gjetur se GSM dhe 3G/4G, një fraksion i madh i energjisë është shpenzuar në gjendjet me fuqi të lartë energjie pas çdo kompletimi/ transferimi tipik të dhënash. Kjo energji është referuar si një energji “Tail” / bisht. Bazuar në gjetjet e tyre, autorët zhvillojnë “tailender” si një protokoll për planifikimin (schedule) të transferimit të të dhënave në aplikimet më të përdorshme në mënyrë që energjia e konsumuar të reduktohet. Komponenti bishtit/tail është më i rëndësishëm dhe është shkaktuar nga kohëzuesit e pasivitetit të përcaktuar statikisht nga operatorët e rrjetit. Ndërfaqja WiFi shkakton një fuqi të lartë fillimi për shkak të lidhjes me AP. Ndërfaqja WiFi në telefona përdor teknikën PSM (menaxhim i ruajtjes së fuqisë) kështu që kostoja e mirëmbajtjes (fuqia) është e vogël. Po ashtu bazuar në rrjetin 2G e matje të shumta të bëra në lidhje me konsumin e fuqisë, ndërfaqja WiFi është e vogël kur jemi në gjendjen Idle dhe në gjendje Aktive (të transferimit të të dhënave) por që preket nga transferimi me shpejtësi të ulët ç’ka sjell kohë më të gjatë. WiFi bëhet joeficent për transferime të dhënash me përmasa të vogla, krahasuar me GSM. Në kushte eksperimentale për studiuesit Balasubramanian në [7] WiFi konsumon 1/6 e energjisë

3G dhe 1/3 e GSMs. Me rritjen e madhësisë së të dhënave, eficenca e përdorimit të WiFi është më e madhe. Balasubramanian et al. [7] ka modeluar konsumin e energjisë së 3G dhe WiFi duke përdorur modelin e regresionit linear bazuar në të dhënat mbi matjet në pjesën nga shkarkimet e të dhënave. Duke pasur parasysh sasinë e të dhënave që do të dërgohen në grup/burst, modeli i tyre linear siguron energjinë e konsumuar. Ky model i thjeshtë nuk kap ndikimin e modelit / fluksit të të dhënave në konsumin e energjisë dhe është i varur nga matjet e tyre (p.sh., pajisje dhe parametra të rrjetit).

Huang et al. [41] ka krahasuar energjinë dhe karakteristikat e shpejtësisë për WiFi, 3G dhe LTE duke përdorur modele të matjes bazuar në një telefon të vetëm. Modelet përfshijnë një kurbë lineare të shpejtësisë së të dhënave – energjisë së matur në telefon përderisa fuqia LTE rritet ndjeshëm me normën (rate) të të dhënave. Tregojnë konsumin e energjisë për trafikun e gjeneruar kur ekrani i pajisjes është i fikur në rrjetet 3G dhe 4G dhe propozojnë një kombinim të përgjumjes së shpejt dhe teknikës "batching" për të reduktuar konsumin e energjisë sfond / background.

Perrucci et al. [1] e fokuson studimin e tij në diferencën e energjisë së konsumuar midis 2 rrjeteve 2G e 3G në lidhje me një përdorim të zakonshëm, si SMS, biseda dhe trafik të dhënash. Një studim tjetër nga Perrucci et al. [1, 5] paraqet rezultatet e matjeve që janë kryer në mënyrë që të krahasohet energjia e konsumuar e një mesazhi tekst, zëri dhe të dhëna në pajisjen mobile në rrjetat 2G dhe 3G. Pajisja e përdorur për matje ishte një model smartphone Nokia N95 i cili përdor OS Symbian. Rezultatet treguan që mesazhet tekst dhe shërbimet e zërit konsumojnë më pak energji në 2G sesa në 3G, por kur një volum i lartë trafiku të dhënash është shkarkuar, lidhja 3G konsumon më pak energji. Rezultatet lënë të kuptohet se nëse duhet të arrijmë një jetë më të gjatë të baterisë duhet të jemi në rrjetin 2G dhe vetëm nëse duhet të dërgojmë të dhëna kalojmë në rrjetin 3G.

Pering et al. [21] prezanton në punimin e tij CoolSpots një sistem i cili lejon që pajisjet e rrjetave mobile të kalojnë automatikisht midis ndërfaqeve WLAN dhe Bluetooth kur bëhet transmetimi / marrja e të dhënave për të ruajtur energjinë. Puna është bazuar në faktin se ndërsa WLAN është më i shpejtë se Bluetooth, ai gjithashtu konsumon më shumë energji. Në përdorimin e vlerësimeve eksperimentale të sistemit CoolSpot, autorët kanë mundësi të reduktojnë energjinë e konsumuar në nënsistemet wireless me më shumë se 50% sipas analizës së tyre.

Efekti i përdorimit të lidhjeve paralele TCP për konsumin e energjisë në një pajisje mobile është studiuar nga Nurminen at al [27]. Nurminen thekson se "përsa kohë smartphonët po përdoren për navigim në web, emaile, shërbime multimedia dhe aktivitete të tjera të dhënash, manaxhimi i lidhjeve për rrugët me energji efiçente janë të rëndësishme". Ideja në punim është të studiohet nëse përdorimi i lidhjeve paralele TCP, i cili ka treguar të përmirësojë throughput-in, do të përmirësojë gjithashtu eficencën e energjisë. Shkarkimet TCP gjatë bisedave 3G ose thirrjet VoIP ose kur disa lidhje të tjera stream TCP janë aktive në të njëjtën ndërfaqe janë dhënë si shembuj në të cilat lidhjet TCP mund të ruajnë energjinë.

Schulman et al. [23] dhe më shumë kohët e fundit Ding et al. [46] kanë shqyrtuar ndikimin që mbulimi i rrjetit ka në konsumin e energjisë 3G. Matjet e tyre tregojnë se pajisjet mobile konsumojnë më shumë energji kur kemi dërgimin e të dhënave në kushte të dobëta të lidhjeve. Huang et al. [41] ka studiuar gjithashtu konsumin e energjisë në LTE duke përdorur matjet fizike

në një smartphone të lidhur me një rrjet komercial. Wang at al [71] studion energjinë e konsumuar të transferimit të të dhënave nëpër rrjetin EDGE dhe HSPA dhe WLAN. Në fund, rëndësia e mbulimit të rrjetit në konsumin e energjisë është matur nga studimi i Schulman (Bartandr) [23].

Referuar matjeve të CPU, Zhang et al [122] mat marrëdhënien midis fuqisë së konsumuar dhe përdorimit të CPU dhe vlerave të frekuencë-tensionit. Matja është kryer nga një program testues i cili ruan frekuencat e sistemit dhe kontrollon ciklin e punës të një detyre llogaritëse intensive.

Në [28] Kjaergaard et al. ndërton një sistem të emërtuar EnTracked, i cili planifikon duke u përditësuar me bazë vendndodhje (të dhënat e vendndodhjes të rimarra me GPS dhe të dërguar duke përdorur 3G tek një server i ndërlidhur) për të minimizuar energjinë e konsumuar dhe optimizuar performancën. Punimi tregon se përdorimi i intervaleve të gjatë midis updateve të vendndodhjes redukton energjinë e konsumuar, por gjithashtu redukton saktësinë, derisa këmbësori ose një objekt ciklik mund të lëvizë më shpejt në disa minuta. EnTracked kështu përdor vlerësimin dhe parashikimin e kushteve të sistemit dhe mobilitetit në mënyrë që të planifikojë të përditësuar vendndodhjen e cila raportohet se ruan një sasi të konsiderueshme të energjisë.

Matje eksperimentale për studime të konsumit të energjisë në pajisje të lëvizshme UE të dyja: si për SW të ndërtuar në pajisje të lëvizshme si Nokia Energy Profiler (NEP) apo punimet për autorët [24, 37-39, 56-58, 61, 66-67, 71] ose matës të energjisë të jashtme si HW [78,].

- Në lidhje me kompresimin e të dhënave dhe ekranin:

Transmetimi i një biti të vetëm në një ndërfaqe rrjeti ajrore mund të konsumojë më shumë se 1000 herë energjinë e krahasuar kjo me një veprim llogaritjeje 32–bit [43]. Bazuar në këtë fakt, kompresimi i të dhënave që duhet të transmetohet duket të jetë potencial në kuptimin për reduktimin e energjisë së konsumuar në pajisjet mobile. Tre studimet e autorëve te [43 - 45], të gjitha japin kontribut në këtë subjekt. Shumë algoritme kompresimi të dhënash kërkojnë për më shumë llogaritje, një akses të madh të memories, e cila mund të rrisë shumë energjinë totale të konsumimit për procesin e kompresimit. Kështu që procesi mund të rezultojë në shtim të konsumit të energjisë në vend të reduktimit të tij nëse kompresimi përdoret pakujdes.

Ekranet e pajisjeve smartphone janë bërë më të mëdhenj e me rezolucion më të lartë, duke gjeneruar satisfaksion të lartë të përdoruesve për videot, shkarkimet në web, navigimet, lojërat dhe funksione të tjera. Por rritja në madhësi dhe rezolucion ka rritur gjithashtu konsumin e energjisë së ekranit kështu që ekranet në kohën e sotme janë një konsumues i ndjeshëm i energjisë në një smartphonë. Në punimin sipas autorëve të [42, 48], ata gjithashtu tregojnë së konsumi i energjisë i një ekrani LCD është i ndjeshëm në pajisjet mobile. *Ata tregojnë së buferi (memorja) e frameve të ekranit dhe "buset" janë konsumuesit kryesorë të energjisë dhe konsumi i energjisë është proporcional me framet e aksesit të buferit të ekranit të këtyre komponentëve gjatë një operimi të shfletosjes së ekranit*. Kështu reduktimi i frameve të aksesimit të buferave redukton dhe konsumin e energjisë së ekranit.

- Në lidhje me "computation offloading":

Ideja bazë e "computation offloading" është e njëjtë me komunikimin me nyje "proxy" që është përshkruar shpesh: duke lëvizur ngarkesën nga një pajisje mobile te një njësi më e fuqishme në mënyrë që të ruhet energjia në pajisjet mobile. Në punimin e autorëve Kemp et al. [52] ideja e "computation offloading" për smartphonët është përpunuar dhe prezantohet një dizenjim sistemi praktik. Sistemi i "offloading" është vlerësuar në dy aplikime smartphone: ku i pari është një aplikim analize i përmbajtjes multimedia që performon njohje të objekteve të imazheve dhe i dyti është një lojë e shpërndarë rrjeti që gjithashtu përfshin një vlerësim të objektit. Të dy aplikimet kërkojnë llogaritje të mëdha dhe në përdorimin e llogaritjeve me "offloading", autorët bëjnë të mundur të reduktojnë ndjeshëm konsumin e baterisë (dhe rrisin shpejtësinë llogaritëse). "Computation offloading" është në këtë mënyrë një ndër metodat më premtuese për reduktimin e mëtejshëm të konsumit të fuqisë së pajisjeve mobile. Pra, ideja në këto zgjidhje është bazuar në shkarkimin e llogaritjeve/veprimeve të pajisjes mobile te një infrastrukturë (PC, Server) në distancë në mënyrë që të reduktojmë konsumin e energjisë.

- Në lidhje me kohëzuesit e pasivitetit dhe teknikën e "Batching & Piggybacking":

Kjo është një fushë pranë studimit tonë dhe një ndihmesë të madhe kanë dhënë dhe studimet si më poshtë:

Algoritmi TailEnder i autorëve te [7] punëson një algoritëm zgjedhjeje trafiku duke përdorur afatet përdorues-specifikuara, dhe parashikon aksesin e të dhënave në të ardhmen për të bërë prefetch të dhënat, për të reduktuar transfertat e rrjetit. Çdo pakete i është caktuar një afat dhe është vendosur në buffer. Kur afati i ndonjë prej paketave është gati të përfundojë, të gjithë të dhënat në buffer janë dërguar në grup/njëherazi. Qian et al. [3, 49] ka propozuar një model gjendje makinë 3G për të studiuar ndikimin e kohëzuesve të pasivitetit në konsumin e energjisë duke përdorur një "dataset" të madh trafiku.

Oliveira et al. [50] përdorin një model të statusit-gjendjes për modelimin e okupimit të bufferit te 3G, si rrjedhojë një vlerë fikse e të dhënave (kbps) por pa kryer ndonjë vlefshmëri të modelit të tyre. Vergara et al [55] zgjedhin transfertat e të dhënave duke patur parasysh kohëzuesit e pasivitetit dhe të dhënat e bufferave të RLC në kombinim. Vlerësimi i tyre tregon një përmirësim mbi TailEnder. Në krahasim me veprat e mësipërme, CLBB duket si i vetmi algoritëm që merr parasysh kufijtë e bufferave RLC kur zgjidhen transmetimet e të dhënave për të shmangur tranzicionet e panevojshme të gjendjeve. Punimi në këtë drejtim ka qenë dhe një ndër pikësynimet tona në këtë dizertacion.

Trafik "backfilling"/rimbushje përdor llogjikën e "*piggybacking"* që t'i japë prioritet transmetimit të të dhënave të vetë sfondit gjatë gropave energjitike midis Tails/bishatave të trafikut interaktiv. Kjo mund të konsiderohet si batching (grup) oportunist.

Autorët e TailTheft at [29] kombinojnë agregimin e trafikut dhe përgjumësit e shpejt / FD. Puna e tyre tregon një përmirësim në prefetching e përmbajtjeve duke kryer parashikime më të sakta të përdorimit. Schulmanet al. [23] zgjedh transmetimin e të dhënave në grupime të mëdha në periudha me vlera të mira të sinjalit RSSI.

Authors Calder dhe Marina [35] përdorin intervale të ndryshëm të planifikimit të transferimit të të dhënave. K'on'onen dhe Paakkonen [36, 73] bazojnë algoritmin e tyre në transmetimin e bursteve/grupeve të sinkronizuar duke përputhur kohët e aplikimeve. TailTheft [29] përmirëson funksionin e Tailender dhe përdor FastDormancy FD. FD është një mekanizëm i standartizuar nga 3GPP i cili redukton energjinë e konsumuar të UE duke lëshuar lidhjen nëse nuk ka paketa për të transmetuar.

Në lidhje me konsumin e energjisë në UE, Qian at al. [3, 41] analizon energjinë shtesë të bishtit Tail në rrjetat 2G/3G dhe shpjegon në mënyrë të qartë konceptin e kohëzuesve të pasivitetit të UE dhe kufirit të bufferit të të dhënave RLC të vendosur nga operatorët e rrjetave mobile.

Cinder [53] është një sistem operativ që siguron proceset e nevojshëm për aplikime ose procese me një buxhet të kufizuar të energjisë (rezervë) dhe një kufizim në shkarkimin e shpejtë të energjisë. Kostoja e lartë e dërgimit të të dhënave përmes ndërfaqeve celulare çon në grumbullimin e trafikut përderisa një proces i vetëm ka buxhetin e energjisë jo të mjaftueshëm për të kryer transmetimin i vetëm.

Cotte et al. [47] propozojnë zgjedhje transmetimi të paketave online në bazë të afateve kufi (deadline) për paketë në një mënyrë të ngjashme me TailEnder. Qëllimi është të maksimizojnë vlerën e paketave të dërguar dhe analiza e tyre është e bazuar në algoritme konkuruese (problemi online). Megjithatë, vetëm rezultatet teorike janë paraqitur dhe kursimet e energjisë nuk janë vlerësuar. Koonen et al. [36] ka propozuar përqasjen e kohëzuesve të aplikacioneve të ndryshme për të kryer transmetime "bursty" të sinkronizuara me një periodicitet të caktuar. Calder et al.[35] zgjedh transmetimin e të dhënave të aplikimeve me intervale të ndryshëm të transmetimit në shumëfisha të intervalit më të shkurtër.

Siekkinen et al.[8] propozon për të formësuar trafikun "streaming downlink" duke përdorur një proxy për të krijuar transmetimet "bursty"/të vrullshme. Vlera maksimale e intervalit të ndër-bursteve varet nga sasia e të dhënave të dërguar në *fillimin e shpejtë* dhe madhësinë e bufferit të TCPs së marrësit. Ata kanë analizuar kursimin e energjisë për qasjen e tyre për 3G duke përdorur video streaming dhe streaming audio për LTE. Rezultatet e tyre tregojnë se deri në 60 % kursime janë të arritshme, kur kemi një *streaming muzikë* mbi teknologjinë LTE. Gjithsesi, këto mbeten më së shumti zgjidhje operatorësh, pra në nivel parametrash rrjeti radio.

- Në lidhje me modelet analitike dhe implementimet OS:

Derisa zgjidhja jonë është bazuar në implementimin e një patch/moduli netfilter brenda shtresës kernel të pajisjes mobile, ne përshkruajmë disa nga punimet të cilat implementojnë zgjidhje të ndryshme aplikative apo OS. Në fund ne diskutojmë disa punime në lidhje me matjen e konsumit të energjisë në smartphonë.

Tailender [40] është një teknikë planifikimi/schedule online për pajisje ku përdoruesi ose aplikimi përcaktojnë afatin e caktuar/deadline për transmetimin burst/grupit të të dhënave. Tailender shihet si një mekanizëm planifikimi për trafikun batch (grupim) midis aplikimeve të ndryshëm dhe bën share/ndan energjinë Tail midis tyre. Ideja është për vendosjen e një vlere "timeout" për këto transferime që lejojnë vonesat dhe transferon të dhënat kur koha "timeout" ose transferime të tjera

të ndjeshme me vonesat ka trigeruar një ndryshim të ndërfaqes radio në statusin aktiv. Ka shumë algoritme të ngjashme me Tailender. Fokusimi në teknikat e ruajtjes së energjisë, Vergara dhe Tehrani [55, 69, 72] planifikojnë (schedule) trafikun e të dhënave bazuar në kufirin e bufferit RLC dhe kohëzuesve të joaktivitetit të cilët i llogarisin me anë të një algoritmi tjetër. Kjo është në bazë të njohurive tona e vetmja teknikë që konsideron të dy parametrat e kombinuar. *Njëkohësisht është dhe një udhëzues për atë ç'ka ne duam të implementojmë në këtë punim por duke e shtrire ne teknologjine LTE apo 4G pasi ajo qendron e implementuar vetem ne 3G.*

Në lidhje me kohët e joaktivitetit, Lee et al [34] zhvillon një model analitik për konsumin e energjisë në WCDMA dhe CDMA200 dhe tregon që koha e joaktivitetit duhet të konfigurohet dinamikisht bazuar në sjelljen e përdoruesit dhe kufizimin e baterisë. Por kjo është një parametër rrjeti.

Falaki et al [60] propozon një metodë empirike duke plotuar CDF-t e kohëve "inter-arrival" të paketave për tracet e mbledhura në komunikimin smartphonë në ndërfaqen radio 3G. Ata gjetën se 95% e kohës inter-arrival janë më të vogla se 4.5 sek, dhe propozojnë vendosjen e kohës së joaktivitetit të një vlere fikse T_1+T_2 =4.5sek. Kjo vlerë është shumë më e vogël se koha 20 sek. e bishtit. Ajo konsumon më pak energji, por nuk ka një mbingarkesë të lartë në sinjalizim. Megjithatë, rialokimi i kanalit do të ndodhë shumë shpesh kështu që pajisja UE shpenzon shumë energji për të kthyer ndërfaqen radio nga gjendja fuqi e ulët në gjendje aktive.

Shumë punë është bërë dhe në lidhje me algoritmet për ruajtjen e energjisë së WiFi si punimi i autorëve te [71]. Në WiFi, koha dhe tranzitimi midis gjendjeve është i neglizhueshëm, por ajo që është më e rëndësishme është përcaktimi dinamik i kohëzgjatjes së sleep (gjumit) kur radio WiFi është "Off". Në këtë gjendje paketat nuk mund të dërgohen por AP mund të jetë në gjendje t'i vendosë në buffer ato dhe problemi këtu është gjetja e kohës së gjumit që siguron që asnjë paketë të mos vonohet (duke përmendur një vonesë maksimale specifike).

Punime të tjera tentojnë të reduktojnë energjinë e konsumuar të prodhuar nga web browsing. Zhao et al. [74] riorganizon sekuencën llogaritëse për ngarkimin e një faqeje web në mënyrë që të performojë së pari transmetimin e ri dhe të drejtojë ndërfaqen 3G në gjendje të fjetur/Idle pas kësaj. Në një tjetër studim, ky autor përdor një makinë virtuale bazuar në një proxy që ndryshon përgjigjet origjinale të një webserveri duke i bërë ato më pak CPU intensive.

Në lidhje me implementimet, modulet kernel apo middleware, ne nuk mund të gjejmë një numër të madh punimesh, por mund të përmendim si ato të autorëve te [31, 51, 69, 75]. HiPAC [68] prezanton një zgjidhje të klasifikimit të performancës së lartë të paketave bazuar në algoritmin multidimensional PCP (problem i klasifikimit të paketave). Mungur et al [31] jep një zgjidhje për të arritur konceptin e lëvizshmërisë fundore end-host bazuar në "GSE/8+8 Locator identity split". Implementon një modul netfiler dhe vë në pritje paketat në planin e përdoruesit ku "copëzimi" i paketave kryhet. Të dyja rastet janë bazuar në module kerneli netfilter, por në ndryshim nga zgjidhja jonë ato performojnë copëzimin e paketave në hapësirën e përdoruesit.

Shuo et al [51] përshkruajnë metodën për të reduktuar porcione të energjisë së konsumuar duke mësuar nga trafiku i gjeneruar dhe duke parashikuar kur një burst trafiku do të fillojë apo mbarojë. Ata zhvillojnë një teknikë për të përcaktuar kur duhet të ndryshojë statusi i ndërfaqes radio nga

Akive në Idle dhe një tjetër për ta ndryshuar mbrapsht nga idle në aktive. Gjithashtu, zgjidhja e tyre vlen për trafikun që lejon një lloj vonese pra për atë background ku përfitimi është në rreth 60%. Në llogjikën e tyre ata përpiqen të krijojnë një llogjikë përdorues – server brenda të njëjtit UE. Për këtë krijojnë një modul kontrolli në shtresën aplikative që ndërvepron me shtresën e modifikuar të socket (plainsocketimpl.java duke implementuar dy algoritme) nëpërmjet thirrjeve socket të sistemit. Në këtë mënyrë menaxhojnë statusin e ndërfaqes UE për një transmetim eficent.

Anannd et al [89] propozon një zgjidhje të transmetimeve data nëpër shumë ndërfaqe rrjeti si 3G, 4G e WiFi duke shfrytëzuar mundësitë e realizimit të implementimit në stakun e rrjetit si TCP, socket apo shtresa e protokollit HTTP. Propozimi i tyre është për modifikimn e librarisë HTTP në platformën Android apo AOSP. Modifikimi lejon që në librarinë HttpUrlConnection të lejojnë e mirëmbajnë në mënyrë paralele shumë lidhje të hapura/ aktive dhe përdori sa më shumë ndërfaqet aktive në çdo kohë për një shkarkim ndërsa e ruajnë aktive ndërfaqen me aplikimet.

Pra, ka shumë punime në këtë fushë si në këndvështrimin HW dhe atë SW në smartphonë nga shumë autorë. Në interesin tonë është pjesa SW e zgjidhjeve. Propozime dhe testime në këndvështrimin SW duke i permbledhur janë bërë në:

- Përdorimin e 2G për të dhënat kur kërkohet update për aplikimet aktive në background (2G/3G/4G auto);
- Të zgjidhet ndërfaqja ajrore që është me eficente në energjinë e transmetimit (psh WiFi kur mundet);
- Lidhjet paralele (miks) mund të reduktojnë konsumin e shpejte të E (CS & PS);
- Protokollin TCP mbi UDP, pasi TCP kërkon më pak mesazhe “keep-alive”;
- Mënyrë preferenciale mund të transmetojmë vetëm kur sinjali është i fortë (RSSI);
- Vendosjen në pritje të shumë aplikimeve dhe transferimit të tyre njëkohësisht/ burst.
- Zgjidhje sipas kushteve të kanaleve me më pak konsum energjie;
- Reduktimin e kohëve të mosaktivitetit T_1 e T_2;
- Përdorimin e llogjikës së prefetching dhe offloading në një PC;
- Kompresimin e të dhënave në komunikim;
- Përdorimin e DRX e DTX në komunikime;
- Zgjidhjet e vendorëve të OS.

Në kapitullin vijues flasim për shkaqet apo rrjedhojat e konsumit të shpejtë të energjisë duke shkuar deri në detajet e rrjetit radio si dhe protokolleve të komunikimit.

KAPITULLI IV

4 RRJETI, ELEMENTËT DHE FAKTORËT NDIKUES

4.1 Tipet e shërbimeve në 3G/UMTS

Krahasuar me GSM dhe rrjetat e tjerë ekzistues pa tel, UMTS-i [83 - 87] apo LTE [113] jep një shërbim të ri të rëndësishëm, që lejon negocimin e të dhënave në kanalet radio bartëse. Atributet që përcaktojnë karakteristikat e transferimit mund të përfshijnë shpejtësinë, vonesat e transferimit dhe nivelin e gabimit të të dhënave. Për të qenë një sistem i suksesshëm, UMTS-i si dhe LTE duhet të suportojë një gamë të gjerë aplikimesh që kanë/shfaqin kërkesa për cilësi shërbimi të ndryshme (QoS). Bartësit UMTS e LTE duhet të jenë gjenerik nga natyra, të lejojnë suport të mirë për aplikimet ekzistuese dhe të lehtësojnë evoluimin e aplikimeve të reja. Përderisa shumica e aplikimeve të telekomunikacionit sot për sot janë shërbime në internet, është e natyrshme që këto aplikime dhe shërbime të diktojnë pikësëpari procedurat për mbartjen e bartëseve.

4.1.1 Klasat e shërbimit QoS në UMTS e LTE

Në përgjithësi aplikimet dhe shërbimet mund të ndahen në grupe të ndryshme, në varësi se si ato janë konsideruar ne lidhje me QoS. Për protokollet e reja me komutim paketë, UMTS-i tentoi të përmbushë kërkesat për QoS nga aplikimet apo përdoruesit. Në UMTS e LTE ka 4 klasa trafiku që identifikohen si:

- *Conversational (bisedë)*
- *Streaming, (rrjedhshmëri te dhenash)*
- *Interactive, (ndërveprim direkt)*
- *Background (sfond)*

Dallimi kryesor midis klasave është se sa i ndjeshëm është trafiku nga vonesat: ku klasa *conversational* është kuptuar si trafik me ndjeshmëri më të lartë ndaj vonesave, ndërsa klasa *background* është klasa më pak e ndjeshme nga vonesat. Klasat QoS të UMTS e LTE janë përmbledhur në tabelën 4.1. Klasat *conversational* dhe *streaming* përdoren në transmetimet në kohë reale në rrjetin WCDMA/3G e 4G, ndërsa klasat *interactive* dhe *background* përdoren për transmetime në kohë "joreale" duke dhënë mundësinë për të kryer përzgjedhje/planifikimin të paketave sipas kushteve specifike.

Klasat e trafikut	"Conversational"	"Streaming"	"Interaktive"	"Background"
Karakteristika bazë	Ndikim në kohë	Vazhdimësi	Ndërveprim përdoruesi	I pavaruar
Shembuj të aplikimeve	Biseda ze/video telekonferenca	Video stream	Navigim në internet	Email, ftp, downloade

Tabela 4.1: Klasat e shërbimeve të UMTS e LTE

Klasa *bisedë/conversational* në kohë reale është karakterizuar nga fakti që vonesa fund më fund (E2E) është e ulët dhe trafiku është simetrik apo afërsisht simetrik. Vonesa maksimale fund më fund është nevojshme dhe për perceptimin njerëzor si në rastin e bisedave me video dhe audio, ku nga vlerësime subjektive kanë treguar që vonesat fund më fund të jenë më të vogla se 400 ms për këto raste. Kështu që kushti për vonesat është si detyrueshem.
Klasa *multimedia streaming* është një teknikë për transferimin e të dhënave që mund të procesohen si një stream/rrjedhje e njëtrajtshme dhe e vazhduar. Me streaming, browseri i klientit mund të fillojë të shfaqë të dhëna përpara se një "file" të jetë transmetuar-marrë plotësisht. *Aplikimet streaming* janë shumë asimetrike dhe për këtë pranojnë më shumë vonesa se ato simetrike, sic janë shërbimet e bisedave. Kjo gjithashtu nënkupton që ato tolerojnë më shumë shqetësime (jitter) në transmetim. Shqetësimet/devijimet mund të mbulohen lehtësisht duke përdorur teknikën e buffering.
Trafiku *interaktiv* është klasa tjetër e skemës së komunikimit të të dhënave që është gjerësisht e karakterizuar nga modeli kërkesë - përgjigje nga përdoruesi fundor. Në destinacionin e mesazhit gjendet një njësi e pritjes së mesazhit (me një përgjigje) brenda një kohe të caktuar. Vonesa vajtje-ardhje, apo *rtt* është faktori kryesor këtu. Një karakteristikë tjetër është që përmbajtja e paketave duhet të transferohet në mënyrë transparente (me një BER të ulët).
Klasa *background* ku trafiku i të dhënave si shpërndarja e e-maileve, sms, shkarkimi i databaseve dhe marrja e të dhënave të matjeve mund të shpërndahen e merren në sfond / background derisa aplikime të tilla nuk kërkojnë një aksion/kërkesë të menjëhershme. Vonesa mund të jetë në sekonda, dhjetra sekonda ose deri në minuta. Trafiku *sfond* (*background)* është një nga skemat më klasike të komunikimit të të dhënave që në mënyrë të gjerë karakterizohet nga fakti që destinacioni nuk është në pritje të të dhënave brenda një kohe të caktuar. Pra me pak fjalë është më pak i ndjeshëm nga koha e shpërndarjes. Një karakteristikë tjetër është se përmbajtja e të dhënave nuk kërkon që transferimi të bëhet në mënyrë transparente. Të dhënat që transmetohen duhet të merren pa gabime gjithsesi.

4.2 Njohuri mbi sistemet mobile 3G/UMTS, 4G /LTE

Rrjeti UMTS apo 3G/HSPA [25, 32, 33, 87] është një nga më të përdorshmit në komunikimet *pa tel* sot për sot (dhe mbizotërues për të dhënat). Ai konsiston në 3 nënsisteme bazë: *a)* Pajisja e përdoruesit UE, *b)* nënsistemi i radio aksesit tokësor UMTS (UTRAN), dhe *c)* bërthama e rrjetit (CN). Për ne, në lidhje me studimin tonë interesi është për dy të parat: UE dhe eNB - RNC fusha ku mund të studiohet dhe arrihet një ruajtje e E (energjisë).

4.2.1 Kontrolluesi i radio rrjetit-RNC

RNC-ja (Radio Network Controller) është elementi kryesor në UTRAN. Ka të njëjtin rol me kontrolluesit e stacionit bazë (Base Station Controller - BSC) në rrjetin GSM. RNC-ja menaxhon radio burimet e stacioneve bazë. Një numër i madh i protokolleve që nevojiten për komunikim në mes të RNC-së, eNB dhe pajisjes së perdoruesit janë të realizuara në RNC.

4.2.2 Nb - Nyja B

Node B-ja kryen funksionin e njëjtë që e kryen stacioni bazë për radio transmetim apo BTS (Base Transciever Station) në rrjetin GSM. Node B-ja mund të menaxhojë shumë thirrje e transferime të

dhënash në të njëjtën kohë dhe është e lidhur me RNC-në me anë të ndërfaqes IuB dhe Uu drejt UE apo smartphone-it.

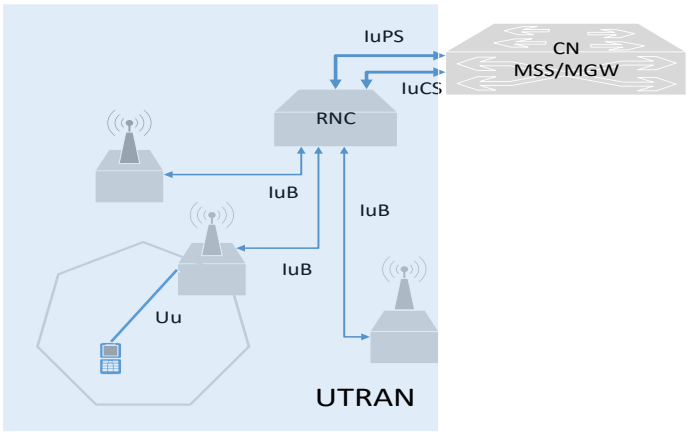


Figura 4.1: Rrjeti i radio aksesit UTRAN në 3G/UMTS

Si për shembull, kontrollimi i fuqisë në qarkun e brendshëm (inner-loop) realizohet në NodeB edhe bëhet me qëllim që të përshtatet vlera e fuqisë së transmetimit në përputhje me vlerën e paracaktuar nga RNC-ja. RNC-ja duhet të jetë e informuar mirë për situatën aktuale nëpër qelizat e rrjetit që janë të lidhura në të, në mënyrë që të mund të marri vendime për handover, kontrollimin e fuqisë dhe pranimin e thirrjeve. Pajisjet UE dhe NodeB-të në mënyrë periodike bëjnë matjen e cilësisë të lidhjes dhe të nivelit të interferencës dhe vlerat e matura ia dërgojnë RNC-s. *Rëndësi të veçantë në studimin tonë merr dhe njohja e protokolleve të komunikimit midis UE dhe RNCs pasi në të specifikohen rregullat e komunikimit. Bazuar në këto specifikime protokollesh në lidhje me komunikimet, implementimi në Android OS mund të realizohet sipas kushteve që ne duam të zbatojmë për përmirësime energjitike*. Zgjidhje të ngashme mund të arrihen dhe në anën e RNCs e eNBs por që kërkojnë vëmendjen e operatorëve apo dhe vendorëve të pajisjeve (për pjesën e parametrizimit apo SW).

4.2.3 UE – pajisja e përdoruesit

Pajisja e përdoruesit apo UE, konsiston në 2 pjesë: ***ME***-pajisja HW/fizike, dhe ***USIM*** – moduli UMTS i identitetit të përdoruesit (SIM per 3G). UE-ja përfshin gjithë pjesën HW (fizike) dhe SW (llogjike apo aplikative) që duhet për shërbimet standarde UMTS /3G apo LTE 4G. USIM mban informacionet specifike të përdoruesit. Struktura e një telefoni smart në përbërjen fizike ka:

- *Nënsistemin celular:* është pjesa e pajisjes përgjegjëse të telefonit për të gjitha veprimet në komunikimin me rrjetin pa tel. Ajo është një njësi dixhitale “baseband” që menaxhon dhe pjesën e modemit 3G (por dhe GSM, GPRS/EDGE, UMTS, LTE, WiFi, Bluetooth apo dhe NFC), aksesit të rrjetit, transmetimin dhe marrjen e sinjalit radio.
- *Nënsistemin aplikativ:* merret me të gjitha algoritmet e procesimeve multimedia dhe ndërfaqet me përdoruesin. Koduesit audio dhe video, kompresimin dhe procesimin e imazhit, komunikimin me gjithë periferikët dhe lidhjet me ndërfaqet e ndryshme me rrjetat (WLAN, Bluetooth and GPS) janë pjesë e kësaj pajisje.
- *Periferikët video:* veçmas njësive procesuese që shpjeguam më sipër, telefonat smart kanë dhe periferikë, të cilët janë të lidhur me module të ndryshme nëpërmjet linkeve

komunikuese dixhital serialë. Këto linke janë të implementuara dhe kontrolluara dhe njëherë nga njësia kryesore e kontrollit p.sh. njësia aplikative. Display/ekrani, ka një peshë kontribuese në gjithë fuqinë totale të konsumuar të sistemit dhe prandaj veçohet në analiza.

- *Nënsistemin e memories:* është një burim kritik i konsumimit të fuqisë duke kontribuar me ~20% të konsumit. Instruksionet e aksesimit të memorjes dhe aksesimi i të dhënave në memorje janë të rëndësisë së parë. Arkitekturat "*pipelining"* janë të mundura për të reduktuar aksesimin e memories dhe tërheqjes së instruksioneve. Teknikat e kompresimit të të dhënave që minimizojnë të dhënat, frekuencën e aksesimit të memorjes, mund të cojnë në reduktim të konsumimit të fuqisë së memories. Kjo qëndron e vërtetë, megjithatë llogaritjet shtesë nuk shtojnë humbje fuqie sesa efekti i ruajtjes së fuqisë që ofrojnë.
- *Ekrani:* është një element i rëndësishëm në pajisjet smartphone. Ka disa modele në varësi të përbërjes fizike. Ekranet IPS–LCD përdorin 2 tranzistorë për çdo piksel. Këto lloj ekranesh mund të konsumojnë më shumë energji se sa një ekran TFT-LCD i cili përdor një transistor për pixel (por kjo sjell ndryshime në rezolucion dhe qartësin në figurë). Ekranet OLED kanë një mënyrë të ndryshme të konsumit të energjisë, ku telefoni ushqen me tension vetëm tranzistorët aktive për zonat nën vëmendje (p.sh. zonat e zeza në ekran për këta tranzistorë nuk u bëhet ushqimi me energji etj.).

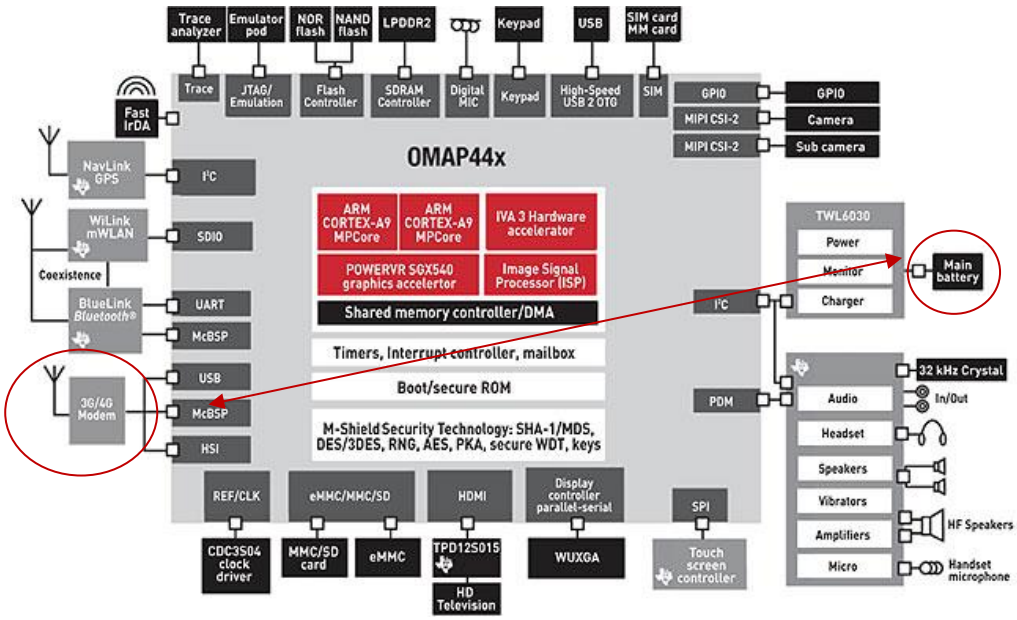


Figura 4.2: Struktura fizike (HW) e një telefoni Smartphone [76]

Figura mësipër jep një arkitekturë të një smartphone me bazë procesor OMAP44 nga ku dallojmë një shumëllojshmëri ndërfaqesh për komunikim e funksione të tjera. Larmishmëria e tyre rrit presionin mbi bateritë e UE.

Studimi ynë është përqendruar në vetëm pjesën e ndërfaqes 3G/4G për UE për një zgjidhje apo planifikim të transmetimeve në kanale specifike për reduktim E. Por dhe shumë punime të tjera sikurse kemi përmendur janë nën studim apo implementuar në ndërfaqe apo për pjesë të tjera HW e SW. Mundësitë e propozuara për përmirësime energjitike shpesh janë përfshirë dhe në release-at e reja SW të OS mobile por dhe në ato RNC e Nb. Kjo mund të ndihmojë masivisht në

smartphone por që dhe shpesh zgjidhjet mund të shkaktojnë problematika të tjera shtesë, prandaj mbetet gjithmonë një punë nën vëzhgim të vazhdueshëm.

4.2.4 Ndërfaqet dhe Protokollet e komunikimit të të dhënave të përdoruesit

Sikurse mund të shihet nga arkitektura e rrjetit, UE lidhet me rrjetin nëpërmjet ndërfaqeve specifike dhe secila prej tyre me protokolle të mirëpërcaktuar në funksione. Në interesin tonë, në këtë punim ne jemi për ndërfaqet [99 - 105] dhe protokollet e përfshira për të dhënat (data). Në figurat më poshtë jepen arkitektura e protokolleve për komunikimin për të dhënat e përdoruesit në teknologjinë 3G / UMTS. *Sikurse mund të shihet UE / smartphone-i i gëzon të gjitha protokollet sikurse dhe pjesa Radio apo Core*. Specifikisht sikurse shihet protokolli RRC ekziston si në pjesën e përdoruesit si dhe atë të kontrollit / RNC, kurse protokolli RLC në pjesën e përdoruesit si në UE dhe në pjesën radio apo RNC (kontrolluesin e rrjetit radio). Sikurse kemi përmendur janë pikërisht këto protokolle përgjegjëse për kontrolluesin e linkut radio dhe sasinë e të dhënave / bufferin të cilat do të shpjegohen më specifikisht për ndikimin që ato kanë në komunikim dhe konsumin energjitik.

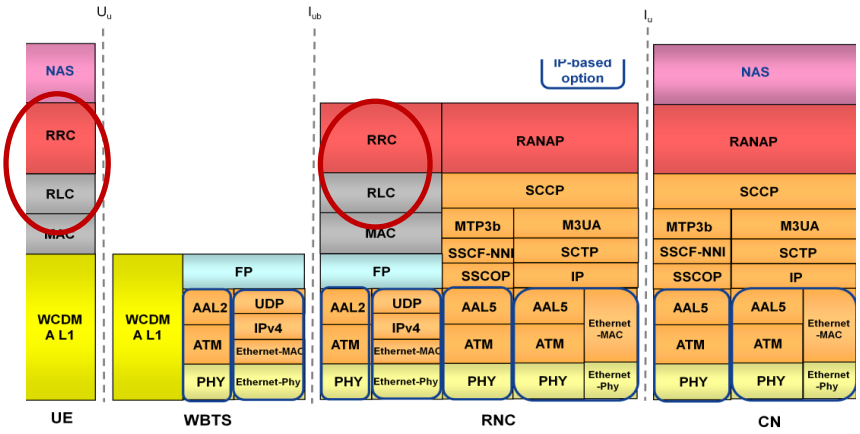


Figura 4.3: Arkitektura e protokolleve të komunikimit UE – CN në 3G (plani i kontrollit)

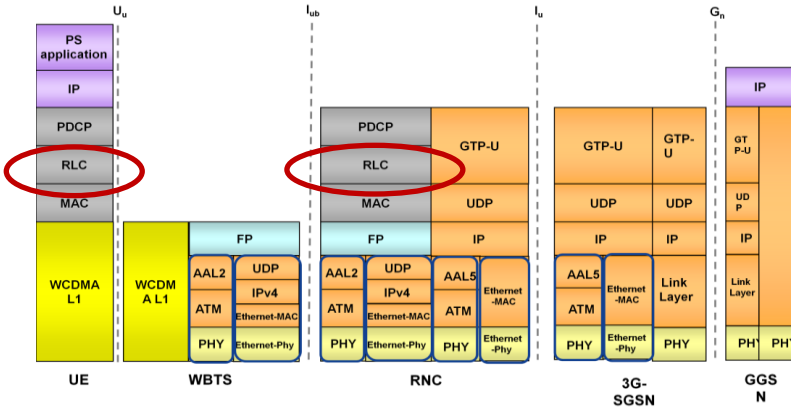


Figura 4.4: Arkitektura e protokolleve të komunikimit UE – CN në IU-PS 3G (plani i përdoruesit)

Ndërfaqja Uu (specifikimi 25.301) është ndërfaqja midis pajisjes *UE* dhe rrjetit të radio aksesit. Ndërfaqja Uu kërkohet për vendosjen, rikonfigurimin, dhe lëshimin e shërbimeve të bartësve radio, përfshirë dhe shërbimet FDD apo TDD. Ndërfaqja Uu konsiston në 3 shtresa:

→*Shtresa Fizike* (shtresa 1)

→*Shtresa e Data linkut* (shtresa e 2 e modelit TCP/IP) që konsiston në 3 nënshtresa:

•*Kontrolli i aksesit të mjedisit* (Medium access control (MAC)).

•*Kontrolli i radio linkut* (Radio link control (RLC)).

Si shtesë, shtresa e data linkut përmban dhe protokollet që i përkasin planit të përdoruesit (user plane):

•*Kontrolli Broadcast/multicast* (BMC).

•*Protokolli i konvergjimit të të dhënave paketë* (Packet data convergence protocol (PDCP)).

→*Shtresa e Rrjetit* (shtresa 3) përmban një protokoll që qëndron në planin e kontrollit:

•*Kontrolli i radio burimeve* (Radio resource control (RRC)).

Figura e protokollit të ndërfaqes Uu përmbledh protokollet e përfshira në ndërfaqen ajrore. Sikurse mund të shihet:

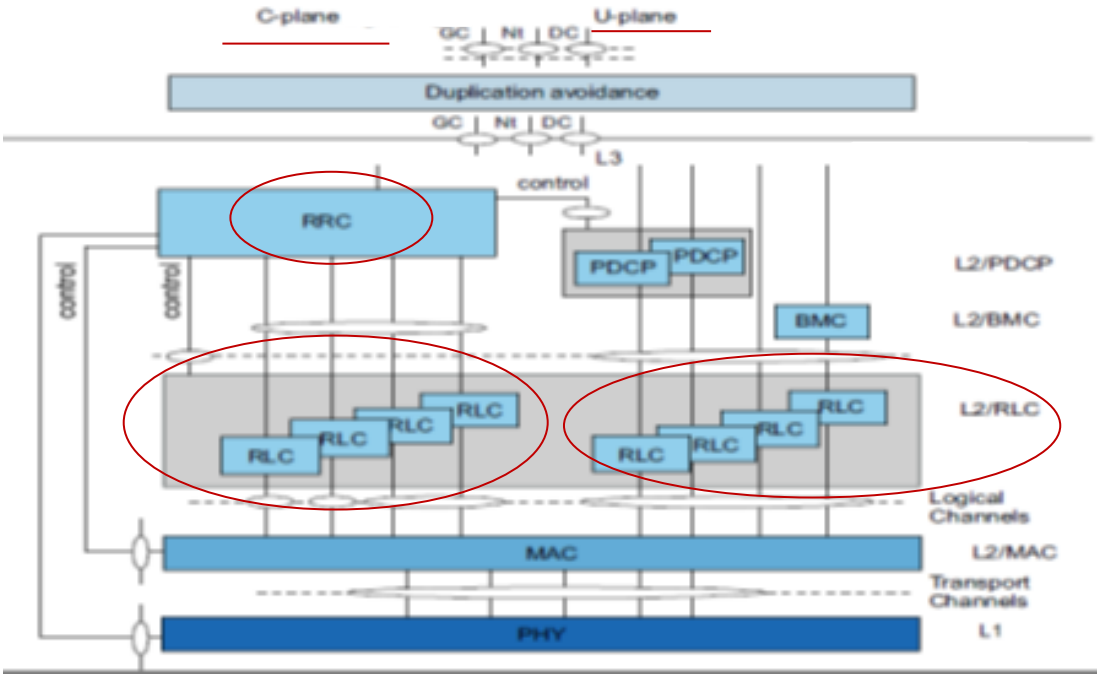


Figura 4.5: Protokollet e ndërfaqes ajrore Uu (UE - NodeB)

RRC ofron funksionet për lidhjet e sinjalizimit drejt / nga UE si dhe funksione të kontrollit të thirrjeve dhe gjendjeve të statusit të komunikimit. Egzistojnë shumë procese dhe nënprocese që mirëmbajnë disa parametra dhe për kontrollin e overload-it në anën e RNC's. Grupi i protokolleve në komunikimin UE – CN (bërthama e rrjetit apo Core) jepet si në figurën 4.4 mësipër.

4.2.5 Kontrolluesi i radio burimeve (RRC në shtresën e 3- të të planit të kontrollit)

Sinjalet e kontrollit konsistojnë më së shumti në mesazhe të kontrollit të burimeve në ndërfaqen Uu. Kontrolli i burimeve radio ofron shërbime te shtresat më sipër nëpërmjet shërbimit të akses point/pikave të aksesit. Pikat shërbyese të aksesit janë përdorur nga protokollet e shtresave me të larta në anën e UE dhe më së shumti nga protokolli Iu Ranap në anën e RAN (rrjetit). Ndërfaqet e kontrollit midis kontrolluesit të radio burimeve dhe të gjithë shtresave të ulëta të protokolleve janë përdorur për:

- Konfigurimin e karakteristikave të shtresave më të ulëta të njësive të protokolleve, duke përfshirë parametrat për kanalet fizike, transportit dhe atyre logjike.
- Komandimi i shtresave më të ulëta për të kryer disa tipe matjesh.
- Raportimi i rezultateve të matjeve dhe gabimeve në kontrollin e burimeve radio.

Për shkak se shtresa e kontrollit të radio burimeve mbart pjesën kryesore të sinjalizimit të kontrollit midis MSS/Core dhe RAN/radio, ajo kryen shumë funksione [64]. Shumica e këtyre funksioneve janë pjesë e algoritmeve të menaxhimit të radio burimeve. Funksionet kryesore të kanaleve të burimeve radio janë:

- Shpërhapja e informacioneve të sistemit.
- Mesazhe paging.
- Zgjedhja dhe rizgjedhja fillestare e qelizës në mënyrën *Idle* në informacionin e sistemit të shpërndarë nga RNC dhe bazuar në matjet (MS-RRC).
- *Stabilizimi, mirëmbajtja dhe lëshmi i lidhjeve RRC midis MS dhe rrjetit të aksesit radio.*
- *Kontrollimi i radio bartëseve, kanaleve të trnasportit, dhe kanaleve fizike.*
- Kontrollim i funksioneve të sigurisë (kodim dhe mbrojtje e integruar).
- Mbrojtje e integruar e mesazheve të sinjalizimit.
- Kontrollimi i matjeve të UE dhe raportimi i matjeve drejt qendrës së monitorimit.
- Funksionet e lëvizshmerise së lidhjeve RRC, sikur një qelize apo update i një zone regjistrimi UTRAN (URA) dhe hondover-at (HO).

Në rrjetin UMTS/3G, një RRC (kontrollues i radio burimit) është pjesë e protokollit që është përgjegjës për caktimin, konfigurimin dhe lëshimin e radio burimeve midis UE dhe UTRAN. Ky protokoll RRC është përshkruar në detaje nga specifikimi 3GPP TS 25.331.

Dy mënyrat bazë që UE mund të jetë janë përcaktuar si: *i qete*/*idle* dhe *i lidhur*/ *utra connected*. Në gjendjen Idle pajisjes UE i është kërkuar një lidhje RRC kurdo që ajo kërkon të dërgojë ndonjë të dhënë përdoruesi ose në përgjigje të ndonje mesazh "paging" kurdo që UTRAN-i apo SGSN-i e bën "page" atë për të marrë të dhëna nga një rrjet të dhënash i jashtëm si psh një "Push" Server. Dy gjendjet përshkruhen më shumë detaje në specifikmin 3GPP TS 25.304 and TS 25.331.

Në figurën 4.6 janë treguar gjendjet e UTRA RRC dhe tranzitimet midis tyre. Kur një UTRA RRC është në mënyrën e lidhur (connected), pajisja mund të jetë në një nga 4 gjendjet sikurse do i shpjegojmë dhe në seksionet në vazhdim.

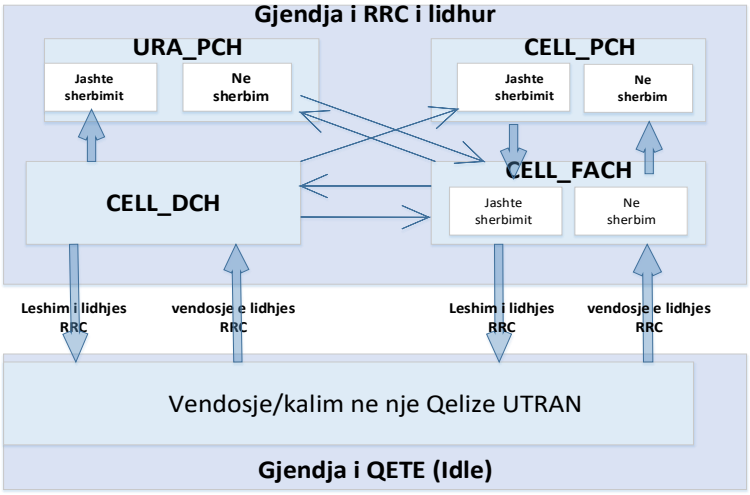


Figura 4.6: Tranzitimi i gjendjeve midis UTRA RRC

Në këtë kontekst aty ka 2 faktorë kryesorë të cilët përcaktojnë konsumin e energjisë si rrjedhojë e aktivitetit te UE_në rrjet. ***E para*** është *energjia e transmetuar*, e cila është proporcionale me gjatësinë dhe nivelin e fuqisë së transmetimit të sinjalit (shtresa fizike). ***Faktori i dytë*** është *protokolli RRC* apo kontrolluesi i radio burimeve [70]. Si rrjedhojë e limiteve të radio burimeve ky protokoll është implementuar dhe në çdo pajisje UE që të përdoret në mënyrë efiçente. Kjo detyrë kryesore e protokollit RRC është të ndërtojë një *"gjendje makine"* në UE dhe ta kontrollojë atë (për sa kemi folur). *Derisa shtresa e rrjetit është ajo ku mirëmbahet informacioni rreth statusit të rrjetit dhe funksioneve për stabilizimin dhe rikonfigurimin e një lidhjeje ekzistuese, këtu është ajo ku reduktimi "i drejtpërdrejë" i energjisë së konsumuar mund të arrihet.*

Shtresa e rrjetit mund të "rikonfigurojë" përdoruesin UE, duke e vendosur atë në gjendje fuqie të ndryshme (në kanale me fuqi të ndryshme). Kur nuk ka të dhëna për të transmetuar, shtresa e rrjetit mund ta vendosë përdoruesin në një gjendje që konsumon më pak energji. Të gjitha këto rregullohen nga protokolli RRC dhe do të merren shumë në konsideratë në këtë punim.

Shtresa fizike ndikon në konsumimin e energjisë në mënyrë indirekte duke shfrytëzuar ndërfaqen ajrore Uu në një kuptim të tillë që të jetë sa më efiçente. Me përdorimin e një modulimi dhe kodimi të duhur, mund të arrihet një kosto më e vogël në terma të fuqisë të transmetuar për bit. *Dizenjimi aktual i RRC duket të jetë i bazuar në një model "ad-hoc" me parametra të konfiguruar në mënyrë statike/fikse.*

Në veçanti ne do fokusohemi në nxjerrjen e kohëzueseve "kritike" të joaktivitetit/pasivitetit të UE që përcaktojnë se kur duhet të lëshojë burimet radio pas një periudhe joaktiviteti/pasiviteti. Gjithashtu do te merremi dhe me protokollin RLC që mirërmban memorjet apo bufferat e brendshëm midis shtresave në komunikim. Si dhe mundësitë e transferimeve me vonesë.

4.3 Kanalet në rrjetin Radio 3G e 4G

Ka tre tipe kanalesh të konceptuara në UTRAN: *llogjike, transporti*, dhe kanale *fizike* si në figurën 4.7. *Kanalet llogjike* qëndrojnë midis shtresës RLC e asaj MAC dhe përcaktojnë se çfarë tipi të dhënash do të transferohen. Këto kanale përcaktojnë shërbimet e transferimit të të dhënave të ofruara nga shtresa MAC (kontrolli i aksesit të mjedisit). Ky është koncepti i kanaleve logjike të përdorura në ndërfaqen mbi atë MAC. *Kanalet e transportit* përcaktojnë si dhe me çfarë

karakteristikash të dhënat transferohen nga shtresa fizike. Këto kanale përdoren në ndërfaqen midis MAC dhe PHY (shtresës fizike). *Kanalet fizike* përcaktojnë saktësisht karakteristikat fizike të radio kanaleve. Këto janë kanalet që përdoren poshtë shtresës fizike, pra në ndërfaqen ajrore. Kjo është pjesa Radio (*pa tel*).

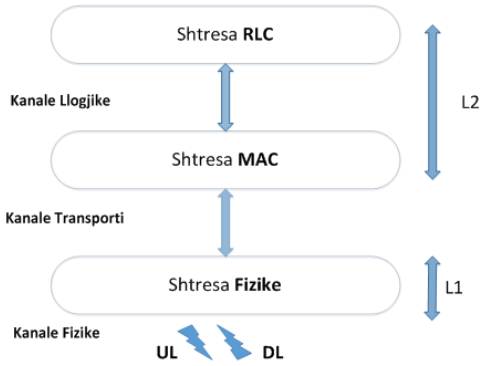


Figura 4.7: Konceptimi i kanaleve në rrjetin UTRAN Radio

Megjithatë kemi gjithsej dy kategori të kanaleve logjike: *a) Kanale kontrolli* e *b) Kanale trafiku*. Ku shkurtimisht:

4.3.1 Kanale kontrolli

Ka gjithsej katër kanale kontrolli në UMTS: **BCCH** (kanali kontrolli broadcast), i cili është një kanal në DL për të shpërndarë informacionin e qelizave të rrjetit. Kanali **PCCH** (kanali i kontrollit paging) që është një kanal DL për të përfunduar paging-un e MS-ve në një qelizë. Kanali **DCCH** (kanali i dedikuar i kontrollit) është një kanal dy-drejtimësh për të vendosur lidhje pikë më pikë midis një UE dhe Utran/RNC. Ai përdoret për proçedurat e RRC (kontrollin e radio burimeve). Kanali **CCCH** (kanali i kontrollit të përbashkët) është gjithashtu një kanal transporti dy-drejtimësh, i cili përdoret për shkëmbimin e informacioneve të kontrollit midis UE dhe UNTRAN.

4.3.2 Kanale trafiku

UMTS-i suporton dy tipe të kanaleve të të dhënave data: **DTCH** (kanali i dedikuar i trafikut) është një kanal komunikimi dydrejtimësh, pikë më pikë i caktuar tek çdo UE për transmetim seri dhe kanali **CTCH** (kanali i trafikut të përbashkët) është i njëjtë me DTCH për transmetime multicast/shumëdrejtimëshe.

Aty ka dy tipe të shpërndara të kanaleve të transportit në përdorim: *a)* kanale të transportit të përbashkët dhe *b)* kanale të transportit të dedikuar.

4.3.3 Kanale të transportit të përbashkët të dedikuar

Kanalet e transportit përcaktojnë si dhe me çfarë karakteristikash të dhënat janë transferuar nga shtresa fizike. Kanalet e transportit janë të ndarë në kanale *të përbashkët* dhe kanale *të dedikuar*. Ato janë të gjitha kanale njëdrejtimshe. Një kanal i dedikuar është i caktuar te një përdorues specifik, ndërsa një kanal i përbashkët mund të përdoret nga të gjithë përdoruesit në një qelizë.

*Kanal i vetëm i dedikuar është **DCH***. Ai përdoret për të transportuar dhe mbajtur të dhënat e përdoruesit dhe sinjalizimit për shtresat më të larta të rrjetit me shpejtësi të lartë. UMTS-i përcakton 6 tipe kanalesh të përbashkët. **BCH** (kanali broadcast) që përdoret për të transmetuar informacionet e qelizës të bëra broadcast nga RNC-ja (kodet e lira scrambling, kodin për kanalin e aksesit të rastit, e kështu me radhë). Gjithashtu ben mapping / përputhjen e kanaleve, që tregon që ky kanal është përdorur nga BCCH. ***BCH** është kanal në vetëm drejtimin DL*.
FACH (kanal i aksesit të përparuar) mbart informacione të kontrollit të përdoruesit. Kanali FACH duhet nga MS-ja për të vendosur/stabilizuar linkun radio. Për këtë ajo përdor një kanal fizik me kod scrambling të madh (shpejtësi të ulët të dhënash, SNR të mirë) dhe është e suportuar nga kontrolli i fuqisë. Kanali *FACH suporton vetëm komunikime DL*. *Kanali **PCH** (kanali paging) është përdorur për të përfunduar proçedurat paging (call setup-in e iniciuar nga RNC).* Kanali suporton vetëm lidhje DL.
Kanali ***RACH** (kanale të aksesit random)* është një kanal UL që së bashku me *FACH* mund të përdoren për vendosjen e një linku radio. Disa operatorë gjithashtu përdorin një kombinim RACH/FACH për të transmetuar një grup të vogël të dhënash. Kanali paketë i përbashkët është një version i zgjeruar i RACH që lejojnë shërbimet PS (komutim paketë) që duhet për të transmetuar shumë frame të njëpasnjëshme. Ai përdoret së bashku me atë FACH. Kanalet suportojnë vetëm komunikime UL. Kanali **DSCH** (kanal DL i përbashkët / shared) mund të përdoret për të transmetuar të dhëna përdoruesi dhe kontrolli që duhet të lexohen nga shumë MS, kanali është vetëm DL. Funksionet janë të njëjta me FACH, por kanali gjithashtu suporton dhe kontrollon fuqinë. Kanali duhet të shoqërohet me një kanal DCH.
Për efekte studimi meqënëse përdoren kanale të ndryshëm midis UL dhe DL për të njëjtat qëllime për transferim të dhënash, do t'i referohemi me të njëjtat emra si PCH, FACH e DCH për të dy drejtimet.

4.4 Përmbajtja PDP

Një kontekst PDP (protokoll të dhënash paketë) ofron një lidhje të dhënash paketë nëpër të cilin UE dhe Rrjeti mund të shkëmbejnë paketa IP. Aktivizimi i përmbajtjes PDP është iniciuar nga UE dhe menaxhon ndryshimin e gjendjes të sesioneve në aktive, krijon një kontekst PDP, merr adresën IP dhe rezervon burimet radio. Pas aktivizimit të një përmbajtje/konteksti PDP, UE është në gjendje të dërgojë e marrë paketa IP nëpërmjet ndërfaqes Air Uu.
UE mund të ketë deri në 11 kontekste/përmbajtje PDP aktive të njëkohshëm, por kjo varet nga implementimet dhe HW/SW të ndryshëm. PDPt përmbajnë informacion për sesionet e përdoruesve kur përdoruesi ndodhet në një gjendje aktive.

4.5 Gjendjet e RRC

Në lidhje me RRC, UE mund të gjendet në një nga gjendjet si në figurën e mëposhtme. Gjendjet janë vendosur në akset x dhe y në përputhje me konsumin e tyre të fuqisë dhe performancën në terma të përgjigjes në kohë dhe shpejtësive të të dhënave maksimale respektivisht:

- Në CELL_DCH është e alokuar për terminalin në të dyja drejtimet UL dhe DL, duke dhënë shpejtësi të lartë të dhënash. Terminali UE ka akses në kanalet e transportit të dedikuar UL ose DL, kanalet e ndarë/share të transportit dhe një kombinim të tyre.

- Në CELL_RACH / FACH terminalit i është caktuar një kanal i përbashkët default ose një kanal transporti i ndarë/share në UL (RACH) dhe monitoron DL (FACH) në mënyrë të vazhdueshme. UE mund të transmetojë paketa të dhënash me shpejtësi të ulët.

- Në CELL_PCH (Paging Channel) dhe URA_PCH (UTRAN Registration Area Paging Channel), këto 2 gjendje lejojnë që UE të kalojë më shpejt në gjendjet apo kanalet e mësipërme. Disa operatorë nuk i implementojnë gjendjet opsionale.

Duhet theksuar se në CELL_DCH dhe CELL_RACH / FACH UE mbetet "i lidhur" me RNC. Kjo ka nje efekt direkt mbi energjinë që UE konsumon pasi ndërfaqja e rrjetit mbetet në status me energji të lartë e të mesme duke cuar në një rrjedhje të shpejtë të konsumit të batersisë.

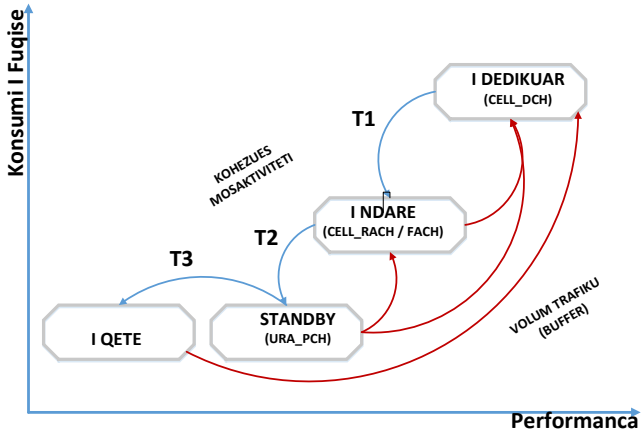


Figura 4.8: Gjendjet e lidhjeve (kanaleve) 3G, performanca dhe konsumi i fuqisë

Tranzitimi i gjendjeve, në UE ndodh bazuar në volumin e trafikut dhe kohëzuesit e joaktivitetit të kontrolluar nga RNC. RRC përdor informacionin nga protokolli RLC në mënyrë që të raportojë volumin e trafikut të transmetuar apo marrë nga rrjeti. Psh, në FACH, UE raporton te RNC volumin e trafikut të vëzhguar bazuar në statusin e të dhënave në buffer. Kjo ndihmon RNC të rivlerësojë alokimin e burimeve. Bufferi i të dhënave RLC është përdorur për të trigeruar tranzitimin e gjendjeve, që mund të jetë në UL ose DL.

Kur përmbajtja e bufferit të të dhënave tejkalon një kufi limit, sinjalizimi korrespondues është shkëmbyer përpara tranzitimit të gjendjeve. Këto buffera janë pastruar kur të dhënat janë transmetuar. Kjo logjikë e RRC do të shpjegohet përsëri dhe do të na ndihmojë më vonë në këtë punim. Kjo do të jetë baza për përcaktimet në formatin e planifikimit të të dhënave në drejtimin UL.

4.6 Teknologjia 4G / LTE-A

Arkitektura e evoluar e sistemit SAE, e diskutuar në Release 8 të 3GPP dhe evoluimet e mëvonshme, propozojnë një arkitekturë "të sheshtë" e shumë të thjeshtur që redukton vonesat dhe përmirëson performancën. Figura mëposhtë shfaq elementët e rrjetit në konfigurimin e arkitekturës. Arkitektura e LTE është e ndarë në 4 nivele kryesore: pajisja e përdoruesit UE,

UTRAN i evoluar (E-UTRAN), rrjeti qendror paketë i evoluar (EPC), dhe shërbimet domain. UE, E-UTRAN dhe EPC së bashku prezantojnë atë që quhet EPS – sistemi i evoluar paketë, ku funksioni kryesorë është të japë lidhjet bazë IP [113].
EPC përmban dy njësi portash logjike (gateway), të emërtuara si porta shërbyese (S-GW) dhe porta të rrjetit të të dhënave (P-GW). S-GW është përgjegjëse për rrugëzimin dhe drejtimin e paketave të të dhënave të përdoruesit nga-tek eNB shërbyese. P-GW ndërfaqëson UE me rrjetat e jashtëm të të dhënave (PDNs) si dhe Internetin dhe kryen shumë funksione IP (alokim adresash, komutim paketash dhe rrugëzim të dhënash).

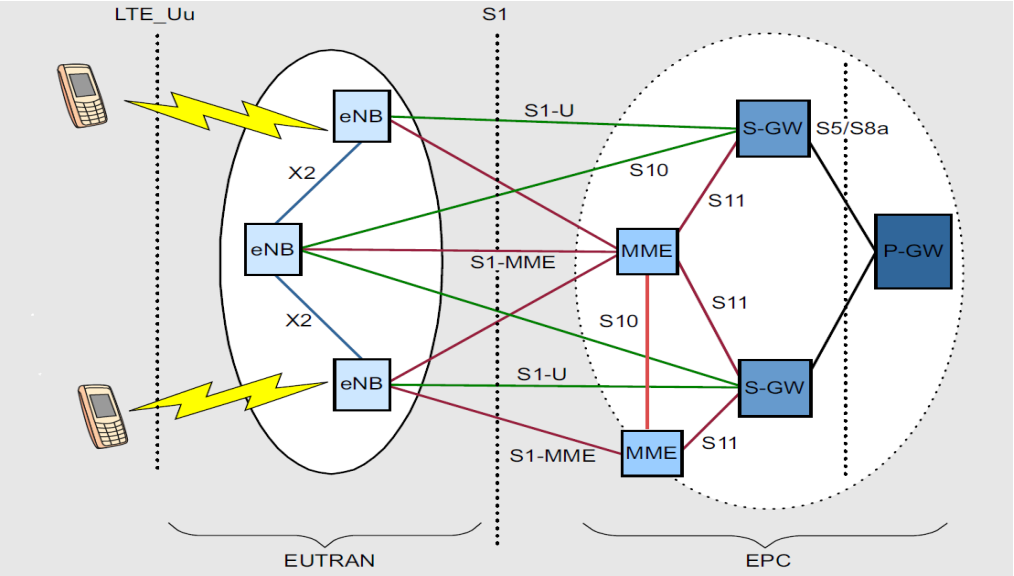


Figura 4.9: Arkitektura e sistemit mobile 4G / LTE-A

Një nga inovacionet më të mëdha të importuar nga SAE është ndarja funksionale midis planit të përdoruesit dhe njësisë së menaxhimit të mobilitetit (MME). MME është vetëm një njësi sinjalizimi, kështu që paketat e përdoruesve nuk janë rrugëzuar nëpërmjet MME. Kështu që është e mundur të menaxhojmë kapacitetin e rrjetit për sinjalizim dhe trafik në mënyrë të pavarur. Funksionet kryesore të MME janë menaxhimi i UEs në gjendje *Qetësie* (Idle Mode), menaxhim të zonave sherbimit, autentikim, romaing, autorizim, zgjedhje e P-GW/S-GW, menaxhimi i bartëseve dhe negocimi i sigurisë.
eNB të ndryshme janë direkt të ndërlidhura në kuptimin e ndërfaqeve X2. E-UTRAN është i lidhur te EPC nëpërmjet ndërfaqes IP. *Ndërkohë në planin e përdoruesit dhe kontrollit kemi po protokollet RRC e RLC me të njëjtat funksione. Pra sikurse shihet dhe në 4G apo LTE përdoret pothuajse i njëjti parim në menaxhimin e gjendjeve*.

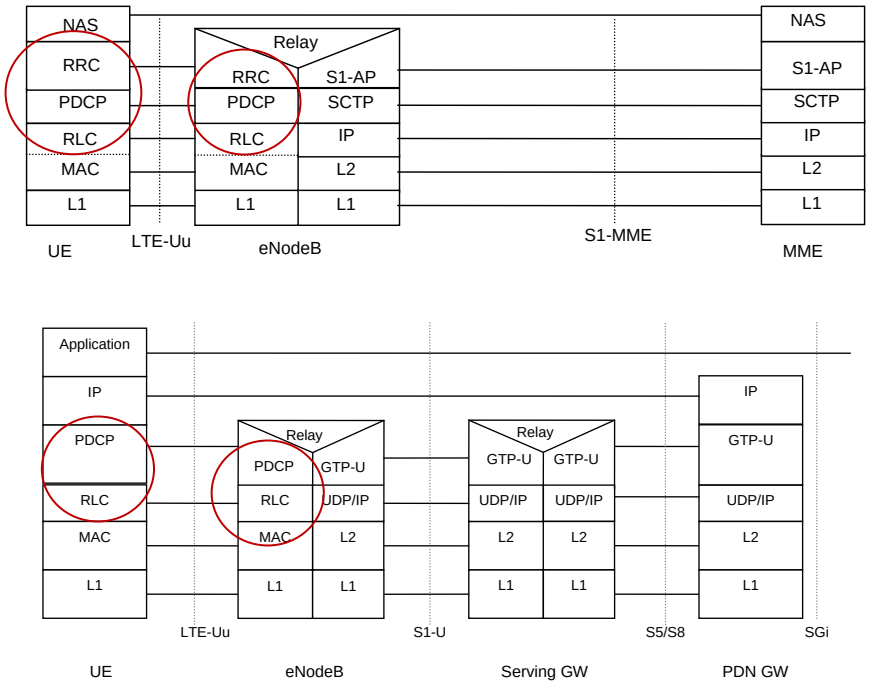


Figura 4.10: Plani kontrollit dhe i përdoruesit në LTE (RRC, RLC)

4.6.1 Kontrolluesi i burimeve radio, RRC në LTE

Mesazhet e kontrollit të radio burimeve paraqesin një pjesë thelbësore të informacionit të kontrollit të shkëmbyer midis UE dhe E-UTRAN. Gjendjet e UE në E-UTRAN janë thjeshtësuar më shumë. Në LTE Release 8 e më pas janë përcaktuar 2 gjendje: *RRC i lidhur* dhe *RRC idle*.

a) Në **RRC Idle / i qete**, UE është i rregjistruar në MME por nuk ka asnjë sesion aktive. Në këtë gjendje UE mund të njoftohet me anë të paging për trafikun DL. UE gjithashtu mund të iniciojë trafik në UL duke kërkuar lidhje RRC me një nyje shërbyese eNB. Për më shumë UE mund të kryejë matje të celave fqinje dhe zgjedhje/rizgjedhje të celave. Në këtë gjendje mobiliteti është i kontrolluar nga UE.

b) Në gjendjen **RRC connected / i lidhur**, UE transferon dhe merr të dhëna tek/nga rrjeti. UE gjithashtu monitoron kanalet e kontrollit të lidhura me DL-SCH për të përcaktuar nëse të dhënat janë të planifikura (scheduled) për të, jep informacione sqaruese të cilësisë të kanalit (CQI), performon matjet e celave fqinje dhe raportimin e matjeve dhe kërkon informacionet e sistemit. Ndryshe nga gjendja RRC e Qetë, mobiliteti kontrollohet nga rrjeti në këtë gjendje. Në figurën mëposhtë jepen gjendjet e manaxhimt të UE në LTE.

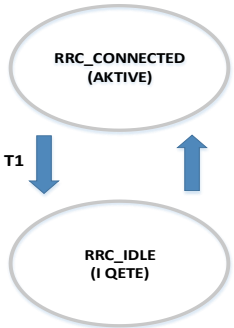


Figura 4.11: Gjendjet RRC në teknologjinë LTE

Komunikimi i të dhënave në rrjet duhet të përshtasë disa funksione për një komunikim sa më eficent dhe fleksibel, dhe për këtë janë krijuar mekanizma të cilat lehtësojnë komunikimet.

Llojet e kanaleve LTE

Ekzistojnë tre kategori në të cilat kanalet e ndryshme të të dhënave mund të grupohen.

- Kanalet fizike: Këto janë kanale transmetimi që mbajnë të dhëna të përdoruesit dhe mesazhe kontrolli.
- Kanalet logjike: Sigurojne shërbime për shtresën e kontrolli i qasjes së mesme (MAC) brenda strukturës së protokollit LTE.
- Kanalet e transportit: Kanalet e transportit të shtresave fizike ofrojnë transferimin e informacionit në kontrollin e qasjes së mesme (MAC) dhe shtresave më të larta. Ne interesin tonë janë këto tip kanalesh dhe për më shumë

Kanalet e transportit LTE

Kanalet e transportit LTE ndryshojnë midis uplink dhe downlink pasi secili ka kërkesa të ndryshme dhe funksionon në një mënyrë të ndryshme. Kanalet e transportit të shtresave fizike ofrojnë transferimin e informacionit në kontrollin e qasjes mesatare (MAC) dhe shtresa më të larta.
Downlink:

- Broadcast Channel (BCH): Hartat e kanalit të transportit LTE në Broadcast Control Channel (BCCH).
- Kanali i Ndarjes Downlink (DL-SCH): Ky kanal transporti është kanali kryesor për transferimin e të dhënave në downlink. Përdoret nga shumë kanale logjike.
- Kanali i Paging (PCH): Për të transmetuar PCCH.
- Multicast Channel (MCH): Ky kanal transporti përdoret për të transmetuar informacionin e MCCH për të vendosur transmetime multicast.

Uplink:

- Uplink Channel Shared (UL-SCH): Ky kanal transporti është kanali kryesor për transferimin e të dhënave uplink. Përdoret nga shumë kanale logjike.
- Kanali i Rastit të Hyrjes (RACH): Ky përdoret për kërkesat e aksesit të rastit.

Koncepti themelor i kanaleve të të dhënave nuk është i ri dhe është përdorur në gjeneratat e mëparshme të sistemeve të telekomunikacionit celular. Kanalet LTE mbajnë shumë ngjashmëri me ato të gjeneratave të mëparshme, por kanalet janë përshtatur për LTE dhe bazohen në funksionalitetin.

4.7 Mekanizmat e komunikimit në rrjet

Në komunikime duhet pasur kujdes në lidhje me eksperiencën e përdoruesit që ajo mos të degradojë nën një kufi të caktuar. Rrjeti apo elemente të ndryshëm si servera etj, komunikojnë me

përdoruesin shumë shpesh. Për të ruajtur një komunikim sa më efikas disa specifikime apo parametra duhet të respektohen në mënyrë që të mos kalohen disa kufij limit. Për këtë rrjetat përdorin kohëzuesa apo mesazhe për shkëmbim njohjeje të cilat lehtësojnë komunikimet.

4.7.1 Keep-Alive (mbaj gjallë)

Web serverat vendosin një vlerë kohe kufi të "mbajtjes-gjallë" në mënyrë që të menaxhojnë lidhjet e aktive aktuale.. Kjo vlerë zakonisht ka vlerat midis 5 – 120 sek si vlerë default për shumë Web servera HTTP. Nëse një klient nuk dërgon të dhëna/kërkesa shtesë brenda periodës kufi limit /timeout, serveri do ta mbyllë lidhjen. Të dy dhe Klienti dhe Serveri kanë mundësinë të rifillojnë një mbyllje të paqme të lidhjes në çdo kohë dhe secili është duke dëgjuar për këtë. Nëse një mbyllje e paqme nuk është dedektuar në rrjet [93-95]. Për më shumë, një server mund të specifikojë se sa kërkesa totale mund të vendosë mbi një lidhje "të përhershme". Kjo mund të jetë e nevojshme për të siguruar që klienti nuk mbetet pambarimisht i lidhur. Për shërbimet që duhet të mbajnë një lidhje të përhershme hapur për një kohë të gjatë, paketat duhet të dërgohen periodikisht brenda një intervali kohe kufi (timeout). Këto paketa të "mbajtjes gjallë"/"keep-alive" janë njohur edhe si "heartbeats", sepse ato lejojnë Serverin ose Klientin të dijë që lidhja është akoma aktive dhe duhet të mbahet hapur.

4.7.2 Efekti ping-pong

Tradicionalisht, një protokoll i "mbajtjes gjallë" përfshin një shkëmbim *Ping-Pong* të mesazheve server-klient. Kjo mundëson serverin të mirëmbajë një informacion në të cilin klientët janë aktivë ose jo. Në mënyrë të ngjashme, një klient mund të dojë të njohë nëse një server është aktiv ose jo. Prandaj për këtë, të dy si klienti ose serveri mund të iniciojnë një shkëmbim mesazhi heartbeat, në varësi të skemës së përdorur. Në përgjithësi, njëra anë dërgon një mesazh *Ping*, i cili përgjigjet me një mesazh *Pong*. Nëse mesazhi Pong është marrë me sukses, ana tjetër e di se nyja tjetër është aktive. Ndryshe, lidhja mund të supozohet joaktive dhe mbyllet. Asnjë nga këto mesazhe nuk ka të dhëna përdoruesi dhe janë të vogla në madhësi. Përsa kohë një shkëmbim mesazhi "heartbeat" është kompletuar, një kohëzues "timeout" fillon përsëri për një cikël tjetër. *Tipikisht, konfigurimi i intervalit dhe frekuencës së "mbajtjes gjallë" është bërë në nivelin e aplikimit. Megjithatë, kjo është implementuar më afër shtresës së transportit për përdorimin me TCP-n.*

4.7.3 TCP Keep-Alive / mbaj në jetë

Gjatë një lidhjeje Idle TCP, nuk ka të dhëna të shkëmbyera mes 2 hosteve. Kjo nënkupton që kur një lidhje TCP është stabilizuar, ajo mund të vendoset në gjendje joaktive për periudha të zgjatura duke tejkaluar në shumë orë, ose ditë. Lidhja do të qendrojë në idle sa më gjatë kur çdo host fundor nuk bën një shkëputje ose restart (shkëput-rilidh), dhe aty nuk ka skema heartbeat-i në nivel aplikimi të aplikuara. Megjithatë, ka periudha kohore kur një server dëshiron të detektojë nëse një klient është i lidhur ose është "rrëzuar"/crashed. Kjo është ajo ku shërbimi i *"mbajtjes gjallë"* është caktuar të përdoret nga serveri i aplikimeve që konsumojnë burimet në interes të një klienti, për të përcaktuar nëse klienti me të vertetë i do këto burime të alokuara [95]. Ai mund të përdoret për të parandaluar shkëputjen si shkak i pasivitetit të rrjetit.

Problematika të TCP (dublikime, ritransmetim etj) hasen në komunikimet e transmetimit të të dhënave. Optimizimi nëpërmjet një parametrizimi sa më të përshtatshëm është një zgjidhje efiçente në dallim nga parametrat default sikurse do të shohim në këtë punim. Të gjitha këto procedura e të tjera të pa përmendura janë disa prej shkaktarëve kryesore që i mbajnë UE't të lidhura vazhdimisht në rrjet dhe krijojnë një konsumim të shpejt të energjisë.

4.8 Mënyrat për zgjidhjen e konsumit të baterisë

Në këtë vrull të zhvillimit mobile (HW dhe SW) dhe problematikave gjithmonë e në rritje në lidhje me konsumin e shpejt të energjisë, mënyrat më të përdorura për zgjidhjen e konsumit të baterisë janë:

- Karikime të shpeshta të telefonit, si metoda më e zakonshme.
- Ofrimi i baterive më të mëdha në mAh (për rrjedhojë rritje të madhësie fizike) kërkon më shumë hapësira në telefon.
- Zgjidhje aplikative / API në lidhje me mënyrën e komunikimit me rrjetin. Por shpesh këto janë drejtuar edhe drejt mënyrave të krijimit të profileve me kufizime në komunikim.
- Përmirësim i parametrave apo kohëzuesve të rrjetit (si në protokollet RRC dhe RLC) dhe UE.
- Zgjidhje HW në telefon apo UE, që janë zgjidhje të vendorëve prodhues HW.
- Zgjidhje SW në softwarët smartphone (si middleware apo patch file – rasti ynë).
- Zgjidhje në mënyrat e komunikimit si "offloading" apo "prefetching".

4.9 Teknologjia 5G

Kjo teknologji është në zhvillim si dhe në 2020 masivisht për implementim globalisht. Eshtë në përfudim të fazës së parë të standartizimit. Do të ofrojë shërbime me shpejtësi të lartë te rendit Gbps e vonesa të shkurtra deri ne 10 ms. Ende nuk ka në treg një gamë të gjerë telefonash inteligjente e të përballueshëm nga masat që ta suportojne atë.

4.10 Propozimi ynë për zgjidhjen energjitike

Duket patur parasysh karakteristikat e komunikimit në modelin TCP/IP si në figurën 4.12 midis aplikimeve dhe servera, paketat në drejtimin UL apo UE drejt rrjetit, dërgojnë "data stream" nga shtresa e aplikimit te ajo TCP e me tej. Më pas detyra e shtresës së 3-të është të fragmentojë në segmente dhe ajo e shtresës së dytë IP të fragmentojë në datagrama. *Pikërisht funksionet e shtresës së dytë për fragmentimin në datagrama dhe mbajtjen e bufferave në komunikim përpara transmetimit në shtresën e aksesit të rrjetit na mundësojnë të implementojmë një zgjidhje SW në këtë shtresë për implementimin e një mekanizmi i cili kontrollon fluksin e datagramave në dy memorje me përmasa të ndryshme dhe të vendosi të transmetojë me një vonesë në mënyrë efiçente në varësi të gjendjes së kanalit dhe memorjes.*

Analizimi i volumit të trafikut, si të dhëna të ulta apo të larta, planifikimi i transmetimeve në mënyrë që për të dhëna të ulta mos të kalojë në kanale specifike si DCH por vetëm në FACH janë metodologji eficente energjie.

1. Utilizim i kanaleve me më pak energji sipas kushteve të trafikut data.
2. Kontroll i sasisë së të dhënave në Uplink për transferim eficence energjie.
3. Vendosje përkohësisht në memorizim i të dhënave dhe trajtim për transmetime me pas.
4. Përdorim i FACH për transmetime të shkurta dhe në background në vend të DCH kur të përmbushen disa kushte vonese (në drejtimin UL) per 3G e 4G.
5. Përdorimi i një kohë vonese prej 60, 90 e 120 sekonda si dhe transmetimi më pas i njëkohshëm i të gjithë të dhënave në background.
6. Ripërsëritje e procesit periodikisht kur këta kohëzues përfundojnë.

Por duhet theksuar se kodimi i të dhënave, zgjedhja e protokolleve në shtresa të ndryshme, shtimi i featurave ose funksionaliteteve të rinj mund të influencojnë dukshëm modelin/strukturën e të dhënave, dhe për rrjedhojë impaktin në konsumin e energjisë. Ndërveprimi i përdoruesit bëhet një faktor kyç kur konsiderojmë strukturën/modelin e të dhënave.

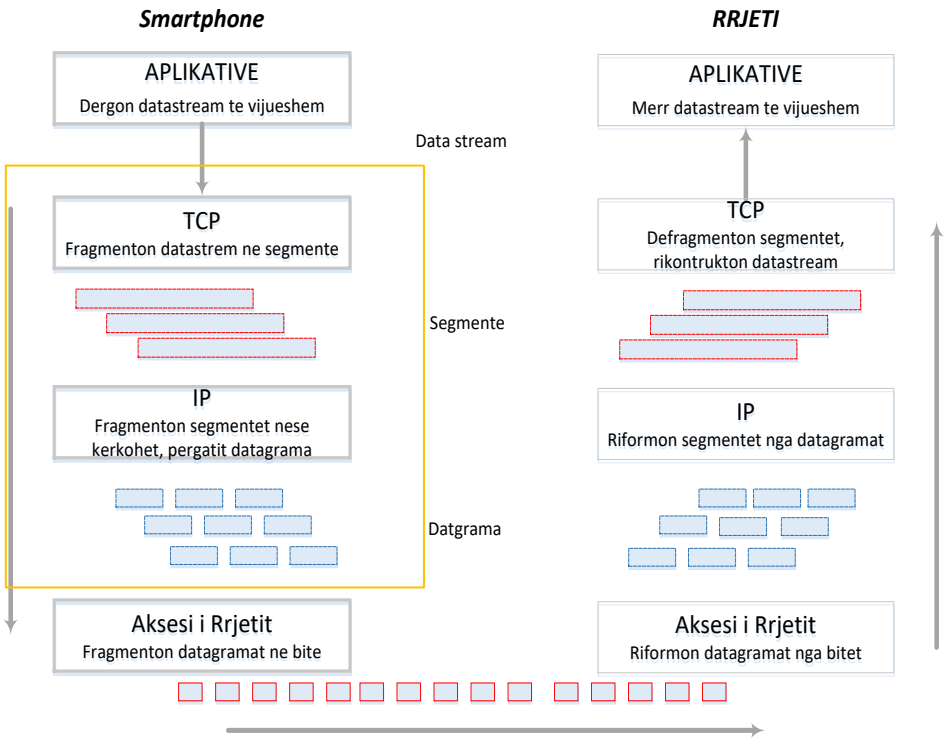


Figura 4.12: Fluksi i komunikimit të të dhënave në modelin TCP/IP

KAPITULLI V

5 GJENDJET E RRC DHE PROBLEMATIKAT ENERGJITIKE NË UE

5.1 Gjendjet RRC dhe implikimet

Ka shumë arsye që ndikojnë në eksperiencën e përdoruesve të pajisjeve smartphone. Dy janë më kryesoret: *kohëzgjatja e baterisë* dhe *eksperienca e vonesave* në shërbime si pasojë e navigimit të shpeshtë në internet. Përdorimi i shpejtësive më të larta duke përdorur teknologji si HSPA dhe HSDPA në lidhjen ngjitëse/uplink dhe lidhjen zbritëse/downlink në rrjetat komerciale, nuk ka ndikuar ndjeshëm në përmirësimin e eksperiencës së vonesave për rrjetet e ngarkuar apo kohëzgjatjes së baterisë. Të dy këta "elementë" kanë ndikim të ndjeshëm me menaxhimin e gjendjes së RRC së përmendur më parë.

Detyra para industrisë së telefonisë mobile është që të kuptohen nga përdoruesi: pajisjet, bateritë e tyre, rrjeti, dhe aplikimet sepse të gjitha së bashku përcaktojnë performancën e duhur për gjithsecilin. Të dyja, si eksperienca e përdoruesit dhe eficenca e rrjetit mund të optimizohen.

Në përgjithësi zhvilluesit e aplikimeve nuk janë të kujdeshëm për karakteristikat specifike të rrjetit celular, si rrjedhojë nga sjellja e aplikimeve të tilla mund të lindin ndërhyrje komplekse. Edhe zhvilluesit më profesional shpesh nuk e kanë vizionin e shqetësimit të burimeve në rrjetat *pa tel*. Situata të tilla në aplikimet "smartphone" jo fort miqësore me rrjetin, rezultojnë p.sh në përdorimin joefektiv të radio kanaleve të rrjetit ose dhe të konsumit të lartë të energjisë në UE dhe mund të vijnë edhe për shkak të mungesës së transparencës në shtresat e ulëta të protokollit të komunikimit. Studimit i nënshtrohet rrjeti UMTS e LTE, si një nga rrjetat më popullor të teknologjisë "pa tel" 3G e 4G, që është implementuar në Shqipëri. Në UMTS faktori kryesor në performancën e aplikimeve dhe eficencës së energjisë së rrjetit është protokolli RRC në UE e rrjet, d.m.th. shfrytëzimi i burimeve radio (të limituara).

Një UE (pajisje celulare inteligjente / smart), mund të jetë në një nga tre gjendjet RRC sikurse kemi përmendur, secila me ndryshim të rëndësishëm të sasisë së burimeve radio të alokuara dhe fuqisë së kërkuar nga ana tjetër. Parametrat e trafikut të aplikacioneve trigerojnë gjendje tranzitimi RRC, të cilat në kthim ndikojnë në shfrytëzimin e burimeve radio, në energjinë e konsumuar të UE, dhe në eksperiencën e përdoruesit. Njohuritë mbi gjendjet e RRCs dhe tranzitimit të tyre janë thelbësore për të siguruar një përdorim efiçent të energjisë së rrjetit [64].

Në thelb dy janë faktorët kryesorë, që përcaktojnë energjinë e konsumuar si rrjedhojë e aktivitetit të rrjetit. Faktori i parë është energjia e transmetuar dhe proporcionet e saj në lidhje me kohëzgjatjen dhe nivelin e fuqisë së transmetuar të sinjalit në shtresën fizike. Faktori i dytë është protokolli RRC, i cili për shkak të limitimeve të radio-burimeve është implementuar dhe në çdo telefon UE, për një punë sa më efektive (këta dy faktorë janë përmendur dhe më parë në këtë punim).

Në rrjetin UMTS apo dhe ne LTE sikurse përmendëm dhe më parë, protokolli RRC ndërton për çdo pajisje mobile një gjendje "makine" për të shfrytëzuar eficencën e burimeve të limituara radio (si p.sh kodet WCDMA). Një gjendje makine RRC mirëmbahet si tek pajisja pa tel UE ashtu edhe tek rrjeti radio apo nyja RNC. Të dyja këto nyje janë gjithmonë të sinkronizuara nëpërmjet kanaleve të kontrollit, me përjashtim të periudhave të përkohshme, tranzitore dhe të situatave ku

shfaqen gabime. Tipikisht janë tre gjendje të RRC në UMTS: *IDLE, Cell_DCH* dhe *Cell_RACH/FACH*. Shpesh në literatura të ndryshme i referohen si IDLE, DCH dhe FACH (duke përmendur dhe përkufizimin për të njëjtin nocion midis kanalesh, midis UL dhe DL).

Gjithsej janë 4 faza të ndryshimit apo tranzitimit të gjendjeve dhe secila prej tyre ka konsum të ndryshëm energjitik:

- CELL_DCH (kanal cele i dedikuar)
- CELL_FACH (kanal cele i aksesit të përparuar)
- URA_PCH ~ CELL_PCH (kanal cele pagging)
- IDLE (gjendje qetësie).

-Gjendja e *Cell_DCH* përdoret kur pajisja e përdoruesit duhet të sigurojë një shpejtësi të lartë dhe vonesa të vogla për transmetim, kryesisht për volume të larta të dhënash ose zëri, p.sh. në rastin e një trafiku video (video streaming) në kohë-reale. Në këtë gjendje pajisja UE ka një kanal të dedikuar si në lidhjen UL dhe atë DL, i cili kërkon një konsum më të lartë fuqie. Një UE mund të aksesojë kanalet HSDPA/HSUPA (High Speed Downlink/Uplink Packet Access), nëqoftëse suportohen nga infrastruktura në gjendjen DCH. Për HSDPA, kanali i transportit me shpejtësi të lartë nuk është i dedikuar, por i bërë share/ndarë nga një numër i limituar përdoruesish (p.sh 32).

-Gjendja e *Cell_FACH* që e ndan (share) kanalin me përdorues të tjerë, përdoret tipikisht në rastet kur ka aktivitet të ulët për transmetime burst (p.sh. në rastin e navigimit në web etj), dhe ku vonesat lejohen. Kjo gjendje kanali shfrytëzon rreth gjysmën e fuqisë në krahasim me gjendjen DCH. FACH është kanal në DL kurse i ngjashëm në UL është RACH.

-Gjendja *URA_PCH*/Cell_PCH përdoret vetëm për të dëgjuar kanalin. Këtu UE përdor DRX (marrje me ndërprerje). Transmetimi në uplink nuk është i lejuar. Kjo gjendje konsumon 1% të fuqisë së konsumuar nga gjendja DCH. UE duhet të performojë proçedurën *CELL UPDATE* pas çdo ri-zgjedhje qelize. UE mund të transmetojë të dhëna përdoruesi vetëm nëpërmjet kanaleve share/të ndarë me shpejtësi të ulët që janë më pak se 15 kbps.

Gjendjet e RRC	Konsumimi i fuqisë (mA)
Cell_PCH / Ura_PCH	<5mA
Cell_FACH	100-150mA
Cell_DCH	200-300mA

Tabela 5.1: Shembull konsumimi i fuqisë për gjendje të ndryshme 3G

Gjendja IDLE tipikisht përdoret në mungesë të ndonjë aktiviteti në rrjet dhe lidhja RRC është e shkëputur. Kjo është gjendja që konsumon më pak energji. Tabela 5.1 jep një shembull të konsumit të fuqisë (rryma e kërkuar) për gjëndje të ndryshme të RRC-së, ndërsa figura 5.1 fakton fuqitë e kërkuara për çdo gjendje nga matje reale në rrjeta (vlerat në *wat*). Kohëzuesit e pasivitetit dhe kufijtë e bufferave RLC ndryshojnë midis operatorëve duke influencuar konsumin e energjisë së UE. Siç kemi thënë gjendja makine RRC është mirëmbajtur nga të dyja, UE dhe RNC.

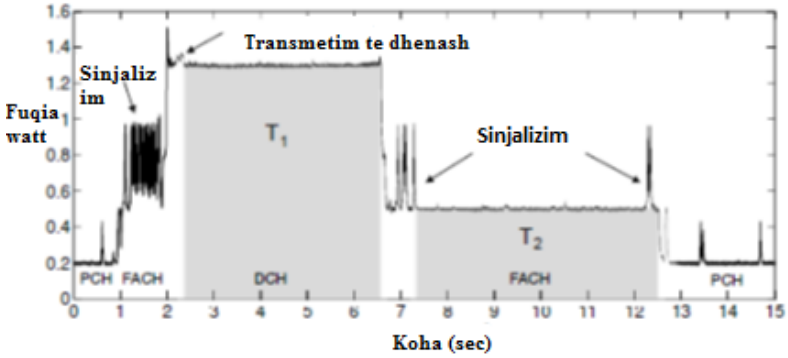


Figura 5.1: Konsumi i energjise në gjendje të ndryshme të ndërfaqes 3G radio

Të dy njësitë (UE – RNC/eNB) janë gjithmonë të sinkronizuara nëpërmjet kanaleve të kontrollit përveç situatave të rastit dhe gabimeve. Gjithashtu duhet thënë se të dyja DL dhe UL përdorin të njëjtën gjendje makine. Një gjendje makine RRC, ka dy tipe kryesore të tranzitimit të cilat do t'i emërtojmë si.

- ***"Promovim"*** i gjendjes: duke përfshirë IDLE→FACH, IDLE→DCH, dhe FACH→DCH, kyçje nga një gjendje me burim radio më të ulët dhe utilizim energjie UE te një gjendje tjetër duke konsumuar më shumë burime dhe energji të UE.
- ***"Demontim"*** i gjendjes: konsiston në kalimet DCH→FACH, FACH→IDLE, dhe DCH→IDLE pra duke shkuar në drejtim të kundërt.

Kur pajisja UE e përdoruesit ndizet, fillimisht ajo qëndron në gjendjen "E Qetë/Idle" derisa të vijë një kërkesë për lidhjen RRC. *Në gjendjen e Qetë/Idle, lidhja e UE është e bllokuar në të gjitha shtresat e rrjetit. Rrjeti Utran nuk ka informacion për gjendjen individuale të UE-së. Në qoftë se UE ose rrjeti Utran ka një paketë për të transmetuar, duhet të vendoset menjëherë lidhja RRC.* Në përgjithësi, Utran-i stabilizon një lidhje të dedikuar RRC dhe qëndron në gjendjen Cell_DCH. Nëse nuk ka diçka për të transmetuar në gjendjen Cell_DCH, bëhet kalimi nga gjendja Cell_DCH në gjendjen Cell_FACH, pas përfundimit të kohës së vendosur në një kohëzues (timer).

Në gjendjen Cell_FACH, pajisja UE merr dhe dërgon paketa të vogla të dhënash në FACH apo RACH. Kur numri i paketave për transmetime kalon një vlerë limit të përcaktuar më parë, statusi i RRC kalon në gjendjen Cell_DCH. Në qoftë se një faqe web e re vizitohet/klikohet në gjendjen Cell_FACH, atëhere tranzitimi në Cell_DCH arrihet në më të shumtën e rasteve (web surfing). Nëse periudha e pasivitetit/joaktivitetit në gjendjen Cell_FACH të RRC kapërcen një prag kohe limit të vendosur nga rrjeti, kalohet në gjendjen URA_PCH. Në këto gjendje pajisja e UE nuk ka ndonjë kanal të lirë të dhënash për përdoruesin dhe aktivitetet uplink janë të pamundura. *Për të transmetuar një paketë të dhënash, është i nevojshëm kalimi në Cell_FACH.* Diferenca midis Cell_PCH dhe Ura_PCH kërkon një ndryshim në proçedurën e "updatimit celës". Pajisja në gjendjen Cell_PCH, UE inicion një procedurë updatimi cele gjatë një ndryshimi cele (ose HO) ndërsa në gjendjen URA_PCH, UE inicion këtë procedurë për një grup qelizash. Në varësi të gjendjes fillestare, një promovim i gjendjes trigerohet ose nga ndonjë aktivitet transmetimi të dhënash përdoruesi nëse UE është i Qetë/IDLE, ose kur madhësia e kërkesës së UE e quajtur madhësia e bufferit RLC tejkalon një limit/threshold në çdo drejtim UL apo DL, dhe kjo nëse UE është në FACH apo IDLE.

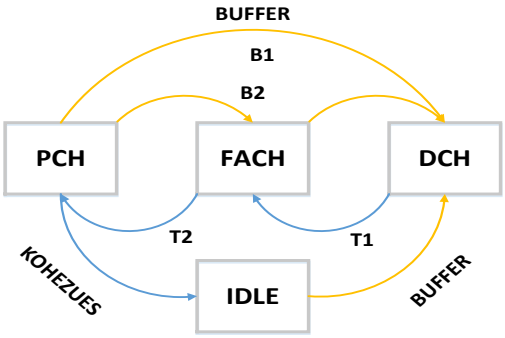


Figura 5.2: Gjendjet e UE dhe kushtet për tranzitimet respektive në 3G

Demontimi / ulja e gjendjeve është trigeruar nga 2 kohëzuesit e mosaktivitetit të mirëmbajtur nga ana e RNC (T1 e T2) sikurse dhe i kemi përmendur më parë. *Në gjendjen DCH, RNC e riseton kohëzuesin T2 në* ***t*** *sekonda (një vlerë fikse), dhe vëzhgon ndonjë frame/kornizë të dhënash DL/UL.* Nëse nuk ka të dhëna transmetimi përdoruesi për T sekonda, kohëzuesi T2 mbaron dhe demontimi i gjendjes bëhet drejt FACH.

Një skemë e ngjashme është përdorur për kohëzuesin ***Tx*** (ku x mund të jetë 1, 2 ose 3) për demontimin e gjendjes FACH→IDLE. *Por duhet përmendur se promovimi i gjendjeve përfshin më shumë punë sesa demontimi.* Promovime të shpeshta të gjendjeve rrisin overheadet e menaxhimit në RNC duke degraduar eksperiencën e përdoruesit specifikisht në rastet për transferime të vogla të të dhënave.

Vlerat ekzakte të kohëzuesve janë të vendosura statistikisht nga operatori i rrjetit në RNC. Kohëzuesit e pasivitetit kontrollojnë tranzitimin e gjendjeve nga shtresat më të larta te ato më të ulëtat. Protokolli RRC mbështetet në informacionin nga protokolli RLC në mënyrë të raportojë volumin e trafikut të vëzhguar në rrjet.

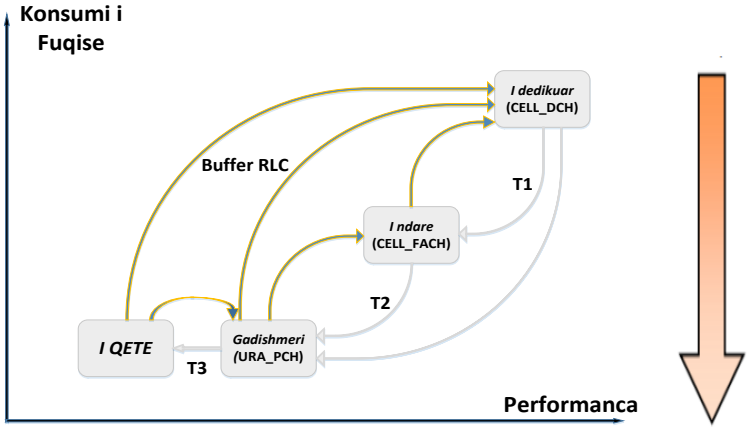


Figura 5.3: Kohëzuesit T1 e T2 dhe nivelet e konsumit të fuqisë për gjendjet RRC

Vlerat e pasivitetit të përdorura në studime të ndryshme tregohen në tabelën 5.2. Respektivisht ndryshojnë nga operatori në operator. *Siç mund të shihet ka 4 memorje/buffera, dy për në UL dhe 2 për në DL, dhe ato janë përdorur për të trigeruar ndryshimin e gjendjeve kur përmbajtja e të dhënave në buffer tejkalon një limit të caktuar.* Ndryshimi i gjendjeve realizohet pasi sinjalizimi korrespondues midis UE dhe RNC kryhet apo MME në rastin e 4G.

Tranzitimi i gjendjes	Madhesia Uplink (Byte)	Madhesia Downlink (Byte)	Vonesa Mesatare e Tranzitimit (Sek)
PCH -> DCH	850 -1000	515	1.7
FACH -> DCH	294	515	0.65
PCH -> FACH	Trigeruar gjithmonë		0.435

Tabela 5.2: Shembull, vlerat e bufferave të të dhënave për ndryshim gjendjeje

Të dhënat e bufferit pastrohen, kur të dhënat transmetohen. Kufijtë e vlerave të përdorura në studim janë matur më parë sikurse shihet dhe në tabelë. Vonesa e tranzitimit është koha që duhet për të realizuar tranzitimin midis dy gjendjeve. Vonesa e promovimit të gjendjes është në mënyrë sinjifikative më e gjatë (të paktën dy herë më shumë) se një RTT normale për të dyja DCH dhe RACH/FACH (më pak se 300 ms). Kjo është e arsyeshme si rrjedhojë e overhead-ve të promovimeve të shpjeguara më parë.

5.2 Variacione të kufijve të bufferave RLC

Sikur shihet në tabelën 5.3 ku korrespondon me probabilitetin e vëzhgimit të një promovimi të gjendjes FACH <-> DCH kur një paketë p.sh prej xx Bytesh (e ndryshme) është dërguar. Në disa shembuj në rrjete mobile është vëzhguar që për DL, kufiri është fiks te 475 Byte ndërsa kufiri i bufferit RLC për UL varion nga 500 deri në 560 Byte. Kjo diferencë është si shkak i ndryshimeve midis kanaleve të transportit DL dhe UL të përdoruar në gjendjen FACH. Tabela mëposhtë jep një shembull sesi ndryshojnë këto vlera midis operatorëve mobile pasi ato janë të konfigurueshme në rrjet.

Kohëzuesi i pasivitetit	Operator 1	Operator 2
T_1:DCH -> FACH T_2:FACH ->Idle	5 sek 12 sek	6 sek 4 sek
Vonesa mesatare e promovimit (në UL)	**Operator 1**	**Operator 2**
IDLE ->FACH IDLE ->DCH FACH->DCH	N/A 2.0 sek 1.5 sek	0.6 sek N/A 1.3 sek
Kufiri i Bufferit RLC	**Operator 1**	**Operator 2**
FACH -> DCH (UL) FACH -> DCH (DL)	540 B 475 B	151 B 119 B
Fuqia mesatare e gjendjes radio	**Operator 1**	**Operator 2**
DCH/FACH/IDLE	800 / 460 /~0 mW	600 / 400 /~0 mW

Tabela 5.3: Shembuj parametrash për 2 operatorë rrjeti mobile

Në rastin e 3G tranzitimi midis gjendjeve DCH, FACH dhe Idle, është i kontrolluar nga kohëzuesit e pasivitetit sikurse është diskutuar dhe më parë. Funksioni i bishtit (Tail) shërben për disa të mira sikurse ai redukton overheadet e kyçjeve pasi në gjendje të lartë përsëri dhe përsëri nëse ekzistojnë transmetime në të ardhmen shumë të afërt.

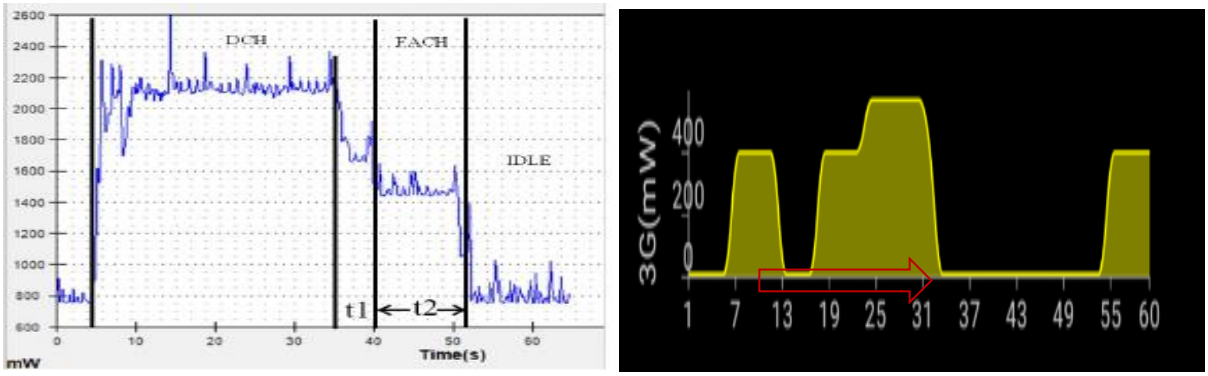


Figura 5.4: Konsumi real i fuqisë në kanal për 3G konsideruar atë bisht T_T[5]

Energjia "Tail" ($T_T = T_1+T_2$) konsumon një energji të konsiderueshme të energjisë totale, ndërsa energjia "ramp" / e ngritjes është e vogël krahasuar me atë Tail (bisht) kjo për shkak të kohës që merr secila. Kur pajisja pret më pak se koha Tail (bisht) për të dërguar paketën tjetër, çdo transferim të dhënash nuk shkakton një energji të veçantë Tail dhe për këtë reduktohet energjia mesatare e transferimit. Kështu që autorët e Tail Ender [3] e kanë dizenjuar në mënyrë të tillë që energjia Tail të mund të utilizohet duke përdorur shumë transferime të njëhershme, por vetëm kjo nëse transferimet ndodhin brenda kohës Tail (bisht) të njëri-tjetrit.

Deng et al [51] mat vlerat e konsumit të energjisë në 3G për shumë aplikime Android sikurse në figurën mëposhtë. Ky grafik tregon përqindjen e energjisë së konsumuar nga gjendje të ndryshme të ndërfaqjes radio. Për shumë nga këto aplikime (të cilat janë aplikime në sfond / background) që mund të gjenerojnë trafik në background pa ndërhyrjen e përdoruesit është verejtur se më pak sesa 30% e energjisë së konsumuar ndodh gjatë transmetimit dhe marrjes së të dhënave, dhe një pjesë e mirë e energjisë rreth 60% është konsumuar nga ndërfaqja 3G kur ajo nuk është duke transmetuar ose marrë të dhëna reale aplikimi (pra vetem gjate tranzitimeve midis kanaleve).

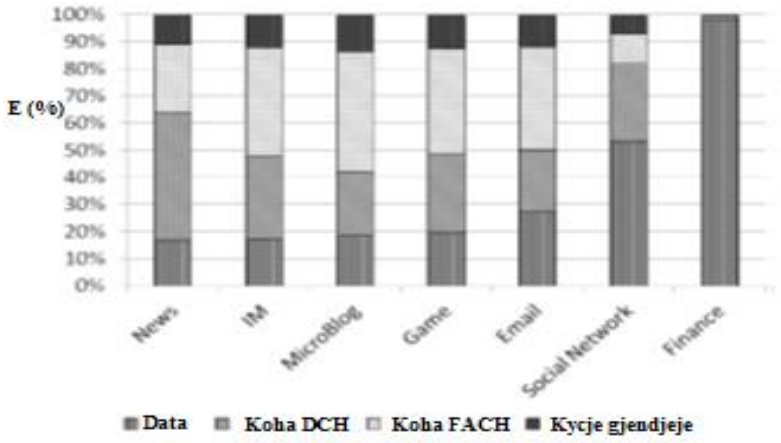


Figura 5.5: Energjia e konsumuar e ndërfaqes 3G për aplikime të ndryshme [51]

Kohëzuesit Tail (bisht) dhe kufijtë e bufferave RLC varen prej operatorëve pasi ato vendosen në mënyrë fikse në RNC. Ndryshimet dhe fluksi i trafikut të të dhënave influencojnë konsumin e energjisë së UE. Transmetimet sporadike të paketave të vogla mund të çojnë në konsum të lartë energjie të UE. Si një ilustrim, dërgimi i një pakete UDP prej 600 bytesh çdo 3 sek gjatë 1 minute

do të çojë në konsum të njëjtë energjie sikur ai i një thirrjeje zë/voice Skype (me një mesatare 19 kB/s): me 90 dhe 92 J (joule) respektivisht, Terhani et al [55]. Njohja e overheade-ve "Tail" dhe kufijve të bufferave RLC mund të çojë në përfitime të UE e cila mund të bëjë të mundur të parashikojmë ose kontrollojmë tranzitimet e gjendjeve në mënyrë që të reduktojmë konsumin e energjisë.

5.3 Problematika energjitike e smartphonëve të sotëm

5.3.1 "Gropa" Energjitike

Fuqia mesatare e konsumuar në një pajisje "pa tel" smartphone ka pësuar një rritje të ndjeshme. Teknologjia e baterive gjithashtu është zhvilluar, por jo me atë shpejtësi që kanë evoluar pajisjet smartphone. Figura 5.6 ilustron hapësirën/pengesën midis sasisë së energjisë *së kërkuar* nga një pajisje smartphone Nokia për 2 ditë përdorimi aktiv dhe shumës së energjisë që mund *të jepet* nga një bateri standarde në përputhje me analizën e brendshme të Nokia-s. Kurba e kuqe në figurë tregon se sa energji kërkohet nga pajisja dhe rritja e pritur në vitet në vazhdim. Energjia e ofruar nga një bateri standard, e përshkruar si në figurë me ngjyrë blu, është qartësisht jo e mjaftueshme për 2 ditë.

Për shkak të hapësirës/ mungesës midis energjisë së kërkuar nga pajisja smartphone dhe energjisë së ofruar nga bateria, jetëgjatësia mesatare e baterisë në pajisjen Smartphone në përdorim aktiv është me pak se 2 ditë dhe në disa raste ajo mund të bëhet dhe një ditë ose më pak. Në smartphon-ët e sotëm konsumi i energjisë është rritur ndjeshëm, e cila nënkupton se ekziston një variacion i lartë në energjinë e konsumuar nga telefoni smart. Në gjendjen i *Qetë/Idle* (StandBy) pa lidhje aktive në Internet, bateria e UE mund të zgjasë me disa ditë. Por nëse një përdorues ka një GPS aktive apo një aplikim harte (maps) të hapur në sfond / background ndërsa po bën navigim në internet (web browsing) dhe po dëgjon muzikë njëkohësisht, bateria mund të bjerë dhe të shkarkohet tërësisht në disa orë. *Duhet theksuar se jetëgjatësia e baterisë është e paparashikueshme përderisa dhe aktiviteti i përdoruesit nuk është i tillë.*

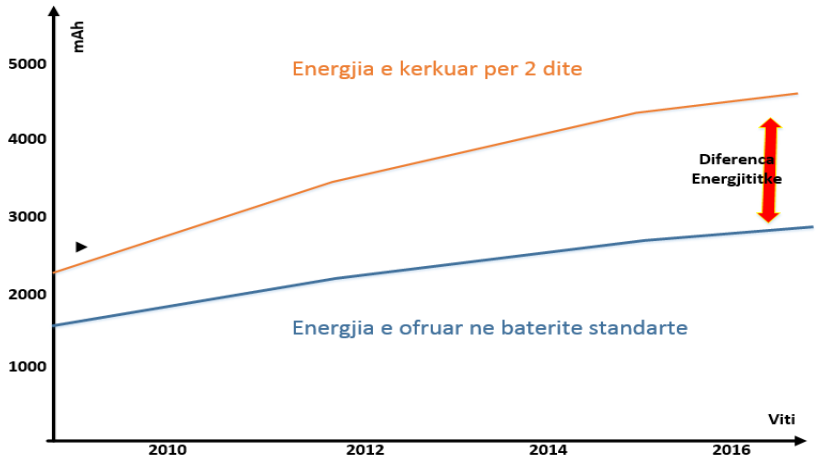


Figura 5.6: Kërkesa për energji e UE përkundrejt energjisë së mundshme [2]

Sikurse Aplikimet *mobile* dhe shërbimet janë dallimi kryesor përgjatë pajisjeve Smartphone nga vendorë të ndryshëm. Në përgjithësi, sa më shumë shërbime të reja që smartphon-ët të ofrojnë, aq më shumë konsum energjie mund të kërkohet. E rëndësishme është dhe koha Idle (e qetë) si një faktor dallues, ku strategjitë për ruajtjen e energjisë po bëhen gjithmonë e më të rëndësishme. Ky nuk është një problem vetëm për përdoruesit por dhe për shërbyesit e shërbimeve mobile (operatorët). Nëqoftëse telefoni nuk ka bateri, përdoruesit nuk aksesojnë shërbimet për kohë të gjatë, duke reduktuar ndjeshëm të ardhurat për operatorët/ofruesit e shërbimeve. Për këtë prodhuesit e pajisjeve smartphone janë të etur për zhvillimin e zgjidhjeve të reja për kohëzgjatjen e jetëgjatësisë së baterisë.

Përdorimi i baterive me më shumë kapacitet mund të jetë një zgjidhje jo fort e pëlqyeshme në lidhje me hapësirat fizike në UE, por fatkeqësisht teknologjia e zhvillimit të tyre nuk ka evoluar sipas ligjit Moore.

"Ndërsa llogaritjet komplekse (procesimet) janë dyfishuar çdo 2 vjet sipas ligjit Moore, kapaciteti i baterisë (teknologjia) është dyfishuar në një dekadë".

Në fushën e baterive, shumica e smartphonëve janë të pajisur me bateritë *lithium-ion*. Këto bateri janë më popullore sepse ofrojnë shumë herë energjinë në fraksione të kohës se sa tipet e tjera të baterive. Për momentin inxhinierët nuk mund të rrisin në mënyrë të ndjeshme sasinë e energjisë së krijuar nga reaksionet kimike dhe e vetmja mënyrë është që: *të krijohen bateri më të fuqishme duke i bërë më të mëdha ato.* Por kjo nuk përshtatet me evoluimin e pajisjeve Smartphone të sotëm të cilët: *tentojnë të kenë më pak hapësirë fizike të lirë për baterinë në mënyrë që të akomodojnë komponenetë për teknologjitë e tjera e të reja.* Por, megjithatë, ka disa tendenca të reja, ku p.sh disa studiues në Universitetin e Standford janë duke përdorur nano-teknologjinë për t'i bërë bateritë të prodhojnë 10 herë më shumë elektricitet sesa bateritë ekzistuese lithium-ion. Kërkues të tjerë në këtë fushë tentojnë të përdorin lëvizjen e përdoruesve për të rikarikuar baterinë, gjithashtu një grup tjetër tenton të inkorporojë karikues diellorë në smartphonë por këto janë vetëm në fillimet e veta dhe do të kërkojnë kohë e kosto ekstra deri në një standartizim.

*Pra, sikurse shihet **Bateritë** janë një komponent shumë i rëndësishëm i smartphonëve të sotëm. Autonomia e smartphonëve varet direkt nga ato, bateritë.*

Ka gjithsej tre tipe baterish të rikarikueshme Lithium Ion (Li-Ion), Nickel Cadmium (NiCd) dhe Nickel Metal Hydride (NiMH). Le të shohim shkurtimisht bateritë e sotme në përdorim [77]. Një pjesë e mirë e informacionit është përfituar nga informacioni në internet.

5.3.2 Bateritë Lithium-ion

Bateritë Lithium janë shumë të përdorshme sot në telefonat celularë/*pa tel* sepse ato janë bateritë e rikarikueshme më energjike sot për sot. Këto lloj baterish u shfaqën kur kërkuesit tentonin të zhvillonin bateri Litiumi të karikueshme. Bateritë Litium–Ion kërkojnë mirëmbajtje të vogël, një avantazh që shumë kimikate (agjentë kimike të përdorur në qelizat e baterisë) nuk mund të pretendojnë. Nuk kanë memorje dhe nuk kërkohet programim ciklik për të rritur jetëgjatësinë e baterisë. Për më shumë ato kanë një shkallë vetëshkarkimi 5-10% për muaj, krahasuar me 30% për

muaj për bateritë Ni-Hy, me përafërsi 1,25% për muaj për vetëshkarkim të ulët të baterive NiMH dhe 10% për muaj në baterinë Ni-Ca.

5.3.3 Kurba e shkarkimit të baterive Lithium-ion

Bateritë Litium-Ion kanë një kurbë shkarkimi të sheshtë ndërsa të tjerat sikurse ato të grafit-acid kanë një pjerrësi të deklaruar. Një kurbë shkarkimi e sheshtë lehtëson projektimin e aplikimeve të cilat përdorin bateritë derisa tensioni ushqyes qëndron konstant edhe pse ka një cikël shkarkimi në vazhdim. Një kurbë e pjerrët ndihmon në përcaktimin e artit të ngarkesës së baterisë derisa tensioni i qelizave të mund të përdoret si një matje për pjesën e mbetur të ngarkesës së baterisë. Qelizat/Bateritë moderne Lithium-jon kanë një kurbë shkarkimi shumë të sheshtë dhe metoda të tjera duhet të përdoren për të përcaktuar gjendjen e *ngarkesës* (charge). Në figurën 5.7 aksi X tregon karateristikat e qelizës të normalizuara si një përqindje e kapacitetit të qelizës kështu që forma e grafikut mund të duket e pavarur nga kapaciteti aktual i qelizave (pjesë përbërëse të një baterie). Nëse aksi X ishte bazuar në kohën e shkarkimit, kohëzgjatja e çdo kurbe shkarkimi duhet të jetë proporcionale me kapacitetin nominal të qelizës.

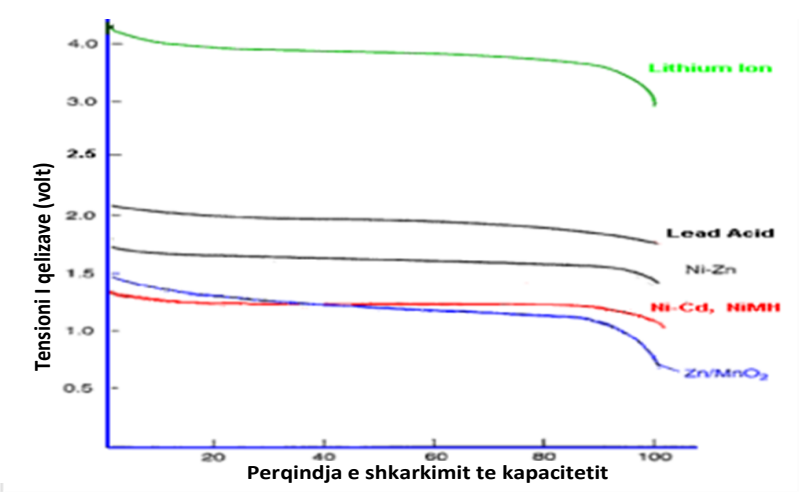


Figura 5.7: Kurba e shkarkimit të baterive me përbërje Lithium-Ion

5.3.4 Karakteristikat e temperaturës

Performanca e qelizave mund të ndryshojë në mënyrë dramatike me temperaturën. Në një ekstrem të ulët elektroliti mund të ngrijë duke vënë një limit të ulët të temperaturës së operimit, ndërsa në ato të larta, ekstreme kimikatet aktive mund të shkatërrohen duke prishur dhe baterinë. Midis këtyre limiteve performanca e qelizave në përgjithësi përmirësohet me temperaturën. Figura 5.8 tregon performancën e baterive Lithium-ion e cila prishet apo “degjeneron” me uljen e temperaturës së operimit. Prandaj dhe kushtet e ambjentit të përdorimit të pajisjeve mobile ndikojnë në energjinë e pajisjeve UE fakt i cili nuk duhet neglizhuar.

I shpjegojmë këto karakteristika sepse me vjetërsimin e baterive bie dhe performanca e tyre dhe testimet në këto kushte devijojnë nga rastet për një smartphone të ri. Ndërkohë dhe ambienti i jashtëm ku ndodhet / qëndron telefoni ndikon në shkarkimin e fuqisë sipas kondicioneve të ambientit.

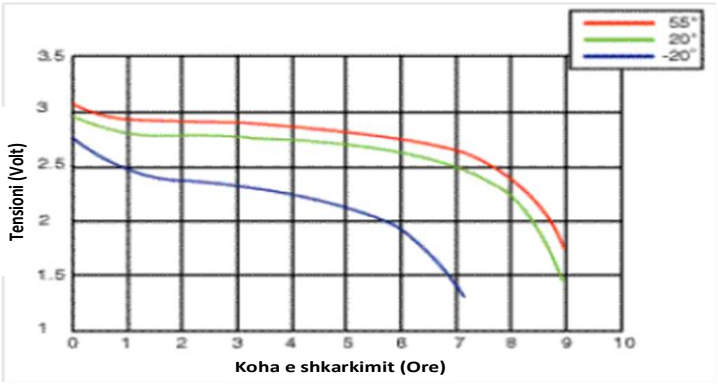


Figura 5.8: Karakteristikat e temperaturës dhe shkarkimit për bateritë

5.4 Përcaktimi Energjitik

5.4.1 Koncepti Energjisë dhe Fuqisë

Eshtë domethënëse të kuptojmë diferencën midis këtyre 2 termave: *Energji* dhe *Fuqi*, të cilat shpesh janë shkëmbyer me njëra tjetrën:

Fuqia: *(shkalla në të cilën puna kryhet)*

Fuqia = Puna = Koha [*Wat*] (5.1)

Energjia: *(koha integrale e Fuqisë)*

Energjia = Fuqia në njësinë kohë [Joules] = P * t (5.2)

Fuqia dhe energjia luajnë një rol kyç në evoluimin e smartphonëve si dhe përmirësimi i performancës së kapacitetit të baterisë është pothuajse shumë i moderuar krahasuar me rritjen e kompleksitetit si shkak i hardware-it të ri dhe shërbimeve të reja. *Përderisa bateria ruan një shumë fikse të energjisë, koha operuese që përdoruesi mund të përdorë telefonin brenda një cikli karikimi, quhet jetëgjatësia e baterisë dhe është fikse gjithashtu***.** Sikurse Smartphon-ët janë duke ofruar më shumë shërbime dhe më shumë aplikime të reja *'energji konsumuese'*, jetëgjatësia e baterisë bëhet më e shkurtër. Për të kaluar këtë problem, energjia e konsumuar duhet të reduktohet. ***Në disa raste reduktimi i fuqisë së konsumuar është i mjaftueshëm për të reduktuar energjinë ekstra të konsumuar në disa raste***. Kjo është e vërtetë për detyrat me një kohëzgjatje konstante (psh video dhe audio, telefonata) sepse energjia e harxhuar është proporcionale me mesataren e fuqisë së konsumuar.

Në anën tjetër, disa detyra të tjera do të kërkojnë më pak energji nëse kryhen më shpejt, por me një fuqi më të lartë sesa me shpejtësi më të ulët dhe me një fuqi të ulët konsumimi por me kohë më të gjatë. Një shembull është shkarkimi dhe ngarkimi i të dhënave. Megjithatë, përdorimi i një fuqie të lartë, edhe për një kohë të shkurtër ka një limit si shkak i nxehjes së pajisjes celulare.

Sikurse kemi përmendur telefonat celulare konsumojnë energji të ndryshme në varësi të mënyrave operuese: *të lidhur* në rrjet apo në gjendje *të qetë* (referuar rastit për të dhënat). Janë bërë shumë

studime dhe shumë testime për të përmirësuar “shterimin” e shpejtë të baterive në smartphon-ët e sotëm. Shumica e tyre konsiston në kushte rrjeti specifik dhe të telefonit smart (gjithashtu duke përfshirë parametrat). Janë propozuar shumë algoritme (në aplikimet mobile), zhvilluar shumë simulime dhe shumë aplikime për të provuar përmirësimin në konsumimin e baterisë (me rezultate të ndryshëm nga njëri-tjetri, pasi dhe kushtet fillestare janë të ndryshëm).

Ne besojmë se një kombinim dhe koordinim më i mirë i parametrave të rrjetit dhe zgjidhjeve OS SW të ruajtjes së energjisë (të maturuar) do të sjellin një përmirësim në jetëgjatësinë e baterisë së telefonave (pavarësisht zhvillimeve në fushën e inxhinierisë kimike per baterite).

5.4.2 Sistemi në këndvështrimin energjitik

Figura 5.9 tregon një ilustrim të thjeshtuar të sistemit ku energjia totale e konsumuar e ndërfaqes së rrjetit wireless është përcaktuar si E(N). E({i}) riparaqet energjinë e konsumuar të një aplikimi në izolim (i vetmi në ekzekutim), dhe E(S) është energjia e konsumuar e një grupi aplikimesh që ekzekutohen së bashku. Kështu, E paraqet karakteristikat e energjisë së konsumuar të sistemit. Këto terma do të përdoren në këtë punim. Disa aspekte ndikojnë në energjinë e konsumuar të një ndërfaqeje rrjeti, duke përfshirë dhe pjesën fizike të UE (si chipset) për një pjesë të saj (psh një grup elementësh elektronikë në qarkun e integruar dhe menaxhimin e tij) dhe komunikimin e të dhënave të krijuara nga sistemi software komandues (psh entitetet). Mekanizmat e punësuar në shtresat e ulëta bëjnë konsumimin e energjisë së komunikimit të të dhënave të varur nga fluksi/modeli i trafikut të të dhënave dhe ndërfaqja wireless.

Kur zhvillojmë një zgjidhje efiçente të energjisë për shërbimet mobile, është e rëndësishme të konsiderojmë që çdo shërbim mund të ekspozojë modele të ndryshëm ndërveprimi dhe prezantojë kërkesa të dallueshme në termat e QoS si ndërveprimi, elasticiteti, toleranca dhe përshtatshmëria [108]. Këto kërkesa limitojnë zonën e zgjidhjes dhe mund të paraqesin kërkesa të ndryshme në terma të kohës, bandwidth-it ose besueshmërisë, të cilat janë matur duke përdorur parametra si jitter, vonesë, përgjigje në kohë, shpejtësia e të dhënave të sistemit ose humbjet [106]. Kërkesat e QoS ndikojnë në modelin e komunikimit të të dhënave të aplikimeve. Kështu, aty nuk ka kërkesa për perspektivën e konsumit të energjisë për klasat e trafikut ose aplikimet mobile dhe energjia e konsumuar është shpesh e konsideruar si pjesë e QoE (Quality of Experience).

Ndërsa QoS (Quality of Service) është një objekt i matshëm për shërbimin edhe për përdoruesin fundor, QoE është shpesh e përcaktuar si një subjekt i qartë matës që riparaqet perspektivën e përdoruesit të shërbimit të matur [112,119]. Në këtë punim, theksi është në prioritizimin e konsumit të energjisë dhe vlerësimin e tij në mënyrë që të mund t’i konsiderojmë të dyja, edhe energjinë edhe dimensionin e QoS, në një mënyrë të kuptueshme.

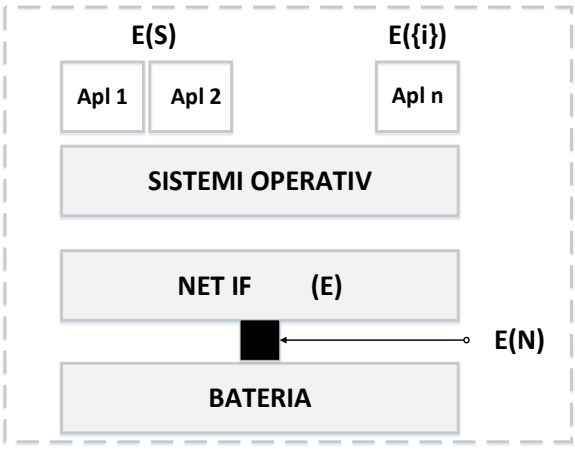


Figura 5.9: Ilustrim i thjeshtuar i energjisë së brendshme të një UE / sistemi

Dy trajtime të përgjithshme mund të përdoren kur zhvillojmë zgjidhje efiçente të komunikimeve mobile:

- **Zgjidhje të përgjithshme për klasa trafiku**: për karakteristikat e trafikut të dhënë, konsumimi i energjisë mund të optimizohet duke njohur kërkesat e QoS (nëse ka). Konsumi i energjisë së një grupi aplikimesh (E(S)) që gjenerojnë një trafik që bie brenda të njëjtës klasë mund të reduktohet nga e njëjta zgjidhje. Një trafik i karakterizuar për të qenë elastik, "përpjekja më e mirë" (best–effort) dhe pa ndonjë kërkesë QoS (përveç faktit që ai duhet të dërgohet) ndërton një fleksibilitet me respektin të optimizojë konsumin e energjisë. Porse optimizimi i energjisë mund të jetë i vështirë për një klasë trafiku "real-time" me kërkesa strikte.
- **Zgjidhje me arnim (Tailor–made)**: janë zgjidhje të besueshme në lidhje me kërkesat e komunikimit për një aplikim. Edhe pse aplikimet mund të gjenerojnë tipe të ndryshme të trafikut, njohja rreth përdorimit specifik të këtyre aplikimeve (psh. karakteristika e utilizimit) mund të sugjerojnë rrugë të mundshme të përmirësimit të eficencës së energjisë. Për shembull, zhvilluesit e aplikimeve do të donin të balanconin kushtin e pandarë midis Performancës – Energjisë në momentin e dizenjimit ose të japin disa nivele përshtatshmërie. Zhvilluesit mund të adoptojnë zgjidhje "tailor-made" për të reduktuar shpenzimin e energjisë të një aplikimi (E({i})) ndërsa mirëmbajnë kërkesat e veçanta për disa aplikime.

Megjithatë, përmirësimi i eficencës së energjisë së një aplikimi ose një klase trafiku për disa aplikime (psh E({i}) dhe E(S)) nuk mund të përmirësojë eficencën e energjisë totale të një sistemi përderisa ndërfaqja wireless është e ndarë ose përdorur nga shumë aplikime. Impakti i ndërveprimit midis aplikimeve të ndryshme dhe ndërfaqes së rrjetit nuk është i neglizhueshëm sepse çdo aplikim kontribon në energjinë totale të konsumuar E(N).

Në këtë këndvështrim, menaxhimi efiçent i energjisë kërkon një kuptim të energjisë së konsumuar të sistemit. Nxjerrja e kontributit të çdo njësie/ entiteti në sistem (p.sh aplikim ose nënproces) te energjia totale e konsumuar është esenciale për menaxhimin e energjisë. Llogaritja e energjisë është metodë që vlerëson sasinë, analizon dhe raporton konsumin e energjisë të njësive/entiteteve të ndryshëm ose aktiviteteve për të dhënë transparencën e duhur të një sistemi. Fatkeqësisht,

përcaktimi i kontributit të çdo entiteti te shuma totale e energjisë së konsumuar të një sistemi ose e një komponenti të vetëm përbën një problem "të shpërndarë". Ky është një problem i vështirë në llogaritje në terma të përgjithshëm dhe nuk ka një zgjidhje të mirë e të vetme për rastet e përgjithshme [107,129]. Brenda fushës sonë të studimit, problemi i shpërndarjes së energjisë është konsideruar analog për të përcaktuar kontributin e çdo aplikimi te konsumi i energjisë totale të ndërfaqes wireless apo asaj të rrjetit. Kryesisht për aplikimet me përdorim protokollet UDP.

5.4.3 Bateritë dhe përcaktimi i kapacitetit të tyre

Bateritë janë pajisje që ruajnë energjinë dhe shumë të përdorshme për furnizimin me energji të pajisjeve portable sikurse telefonat smart, laptopët apo dhe pajisje të tjera argëtuese por gjithashtu dhe pajisje të tjera *mobile* që mund të përdoren në lëvizje. Termi i baterisë i referohet një sistemi prej një apo më shumë qelizash përbërëse. Një qelizë baterie riparaqet një pjesë kombinimi kimik, e mundur për të prodhuar një Tension dhe Rrymë. Kombinime të ndryshme kimike prodhojnë dhe tensione të ndryshëm. *Duke kombinuar qelizat në seri Tensioni i baterisë mund të rritet si shumëfish i tensioneve të numrit të qelizave*. Bateritë e ruajnë energjinë nëpërmjet ndryshimeve të reaksioneve kimike të brendshme.

5.4.3.1 Kapaciteti i baterive

Kapaciteti i baterisë është ngarkesa elektrike, e cila mund të shpërndahet nga bateria në nivelin e caktuar të tensionit (zakonisht referohet si "tension nominal"). Sa më shumë material elektrode të gjendet në qelizë aq më i madh eshte kapaciteti i tij. Një qelizë e vogël ka më pak kapacitet sesa një qelizë më e madhe me të njëjtën permbajtje kimike, megjithëse ata zhvillojnë të njëjtën tension të qarkut të hapur. Kapaciteti i baterive publikohet nga çdo prodhues si vlerë nominale për një grup kondicionesh shkarkimi. Parametrat e performancës së baterive përfshijnë: *Tensionin, Kapacitetin Amp-Ore, Shkallën e shkarkimit (C rate)*.

*Njësitë për matjen e baterive shprehen si **mAh** – miliamper orë. Kjo tregon shkallën e shkarkimit në Amper që një bateri mund të mbajë për një periudhë prej një ore.* Marrëdhënia matematikore për një kapacitetet Q është thjesht produkt i: *rrymës I x kohën t*.

$$Q = I * t \quad (5.3)$$

*kapaciteti në miliamper-ore (Q) = (I) rryma në amper * 1000 * (T) koha në orë*

Kapaciteti i baterisë varion nga shkalla e shkarkimit. Kur shkarkimi i baterisë është i lartë, kapaciteti Amp-orë i baterisë do të jetë më i vogël se kapaciteti nominal i publikuar nga prodhuesi. Sikurse u tha kapaciteti i baterive matet me Ah. Një pjesë që mund të shpërndajë 1 Amp për një orë ka një kapacitet 1Ah. Për ngarkesa konstante, ***llogaritja e kohëzgjatjes së jetës së baterisë është e lehtë, dhe jepet me ekuacionin:***

$$T_{jb} = C / I \quad (5.4)$$

Ku **T_{jb}** është kohëzgjatja e baterisë, **C** është kapaciteti dhe **I** është shkarkimi i rrymës. Megjithatë, si shkak i disa karakteristikave që janë prezente në kushte reale ky ekuacion nuk vlen për bateritë

reale. Këto karakteristika do të diskutohen më tej. Avantazhi relativ dhe disavantazhi i këtyre teknologjive të baterive (në lidhje me kapacitetin, kohën e karikimit, efektin e memories, peshën, fuqinë dhe kostot) janë të njohura mirë për prodhuesit. Njësia matëse e densitetit të energjisë, shprehet në *Wat-orë* për *kg* ose si *Wat-orë* për *litër*. Për bateritë Li-ion është rreth 150-250 Wh/kg (540 – 900 kJ/kg). Zakonisht rekomadohet që bateritë e telefonave të ndrohen cdo 2 vjet për shkak të performancës në degradim në lidhje me energjinë.

5.4.3.2 Nocioni wat për orë

Energjia për orë e ruajtur në bateritë llogaritet me përafërsi duke shumëzuar vlerën e Amp-orë me tensionin:

$$\mathbf{E(J) = P(W) \times t(s)} \quad (5.5)$$

P (Wat orë) = U (Tensioni baterisë) * I (Amp orë)

Pra joules = watts × seconds ose J = W × s

Sa më i madh tensioni i baterisë me një kapacitet të ulët (amp orë) ai mund të shpërndajë të njëjtën energji sikurse një bateri me tension të ulët por me kapacitet të lartë. P.sh vlera të baterisë prej 24V x 8Ah mund të shpërndajë 192 *wat/orë*, ndërsa një model 48 V x 4 Ah gjithashtu mund të shpërndajë 192 *wat/orë*. P.sh. llogaritjen e kapacitetit të baterisë për të llogaritur sa të dhëna mund të marrë për çdo shërbim. Supozojmë që kapaciteti i bateris është 1320 mAh:

*1320 mAh ➔ 1.32 Ah * 3600 s/h = 4752 A s*

Tensioni: 3.7 V

*4752 As * 3.7 V = 17582 W s = 17582 J*

Ne mund të gjejmë se sa Mbyte janë shkarkuar nga programet për çdo cikël të baterisë nëse kemi një eficencë energjie:

*Mbyte të marrë / Cikli i Baterisë = 17582 J * Eficenca e Energjis*ë

Me anë të testeve të matjeve, ne mund të nxjerrim që gjatë një cikli të baterisë telefoni është në gjendje të marrë/dërgojë si një mesatarizim psh:

*Voip: 17582 J * 1.4KB/J ≈24.6 Mbyte*
*Web Browsing: 17582 J * 9KB/J ≈ 160Mbyte*
*Shkarkim File: 17582 J * 80KB/J ≈ 1.4Gbytes*

Duke supozuar që kërkesa për një detyrë, p.sh; shuma e rrymës që një detyrë konsumon nga bateria është e njohur, përcaktimi i kohëzgjatjes së baterisë për këtë detyrë mund të llogaritet duke përdorur këtë ekuacion:

$$\boldsymbol{T}percaktim = \frac{\boldsymbol{C}mAh}{\boldsymbol{I}detyre} \quad (5.6)$$

Kohëzgjatja në orë = 1350 mAh / X rrjedhja e rrymës (mA)

Ku C_{mAh} është niveli i baterisë i informuar nga pajisja UE dhe $I_{detyrë}$ është kërkesa e rrymës për detyrën e kërkuar në mA. P.sh, nëse bateria raporton një nivel prej 900 mAh dhe kërkesa e rrymës për një detyrë është 10 mAh, bateria do të zgjasë 90 orë.

Duke njohur nivelin e karikimit të baterisë, parashikimi i kohës së karikimit mund të llogaritet duke përdorur formulën si vijon:

$$T_{percaktuar} = C_{max} - C_{aktual} / I_{karrikimit} \quad (5.7)$$

ku C_{max} është niveli/kapaciteti maksimali i baterisë së karikuar, dhe $I_{karikim}$ është shpejtësia (rryma) e karikimit të karikuesit të përdorur.

5.4.4 Modeli i automatizuar i fuqisë së baterisë

Gjendja e shkarkimit është përqindja e energjisë së baterisë që është shkarkuar. Të dyja, kapaciteti i energjisë dhe kurba e shkarkimit ndryshojnë me rrymën e shkarkimit, temperaturën dhe "moshën" e baterisë. *Një bateri mund të modelohet si një rezistencë variabël në seri me një burim tensioni variabël sikurse në figurë.* Komponentet e telefonit janë mbajtur në një gjendje të veçantë për një kohë të caktuar dhe ndryshimi në gjendjen e baterisë është përcaktuar duke përdorur sensorë të ndjeshëm të tensionit si pjesë e baterisë për të llogaritur fuqinë e konsumuar për aktivitete. Për të konvertuar tensionet e baterisë në vlera fuqie të konsumuar, variacioni i SOD është llogaritur si:

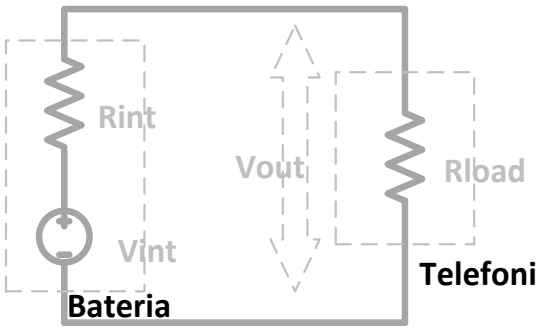


Figura 5.10: Vlerësimi i ngarkesës në smartphone

$$P * (t_1 - t_2) = E * (SOD(V_1) - SOD(V_2)) \quad (5.8)$$

Ku P është fuqia mesatare e konsumuar në intervalin $[t_1, t_2]$, E është kapaciteti i vlerësuar i energjisë së baterisë, dhe $SOD(V_i)$ është shkalla e shkarkimit të baterisë për tensionin V_i (është 1 ose 2). Gjatë vlerësimeve është mirë të vlerësohet shkarkimi i baterisë nga e karikuar totalisht në gjendjen e shkarikimit total duke përdorur një rrymë konstante shkarkimi. Sikurse kemi përmendur edhe për të njëjtin telefon kurba e shkarkimit mund të ndryshojë në lidhje me temperaturën dhe vjetërsinë. Për shkak të rezistencës së brendshme, forma e kurbës së shkarkimit varet nga rryma e shkarkimit. Si rezultat, edhe nëse burimi i brendshëm i tensionit ka një tension konstant, bateria ka të njëjtën "gjendje të shkarkimit SOD", tensioni i pajisjes varet nga rryma e shkarkimit.

Për matje dhe monitorim të Energjisë duhet të përdorim një multimetër për të matur ndryshimin e tensionit V_{diff} përmes rezistorit me precizion të lartë 0.02 Ohm i lidhur në qarkun kryesor të fuqisë 0.02 Ohm. Rryma e çastit mund të llogaritet më pas si:

$$I = \text{Vdiff}/0.02\text{Ohm} \quad (5.9)$$

Kjo metodologji është përdorur në rastin e matjeve të kryera në një smartphone Nokia Lumia 625 si dhe Samsung S3/4 duke matur rënien e tensionit dhe duke nxjerrë rrymën e shkarkimit më pas.

5.5 Metodologji matjesh për nxjerrjen e vlerave në interes

Sjellja dhe performanca e rrjetit UMTS, janë të drejtuara nga një numër parametrash që janë konfiguruar nga operatorët e rrjetave, si psh timeout – "kohe boshe". Në këtë seksion, ne do të tregojmë vlerat aktuale të këtyre parametrave të cilat mund të nxirren apo maten nëpërmjet matjeve fund më fund, pa pasur nevojë për bashkëpunimin e operatorëve të rrjetave. Në parim, një informacion i tillë përdoret nga shumë studiues të rrjetave *mobile* për të studiuar skenarë apo situata të mundshme të rrjetit radio si dhe programues aplikimesh *mobile*. Qëllimi në këtë punim është të tregojmë parimet dhe metodologjinë e matjeve fund më fund për të zbuluar vlerat aktuale të këtyre parametrave dhe në të njëjtën kohë të mbledhim informacione të nevojshme të sjelljes së rrjetit. Fokusi ynë do të jetë në rrjetat mobile UMTS apo teknologjinë 3G si dhe LTE/4G siç njihen ndryshe, që janë ndër teknologjitë më popullore në rrjetat celulare sot në botë.

Kjo gjendje RRC, apo funksion i kontrollit të radio burimeve siç mund të shihet ka ndikim të drejtpërdrejtë dhe vlerësimi i saj nga matjet direkte përbën detyrën në vijim të këtij punimi. Paraprakisht, kemi shpjeguar gjendjet e RRC për UE dhe konsumin e energjisë për çdo gjendje. E rëndësishme është të dimë si mund të kryhen matjet për nxjerrjen e vlerave të kërkuara:

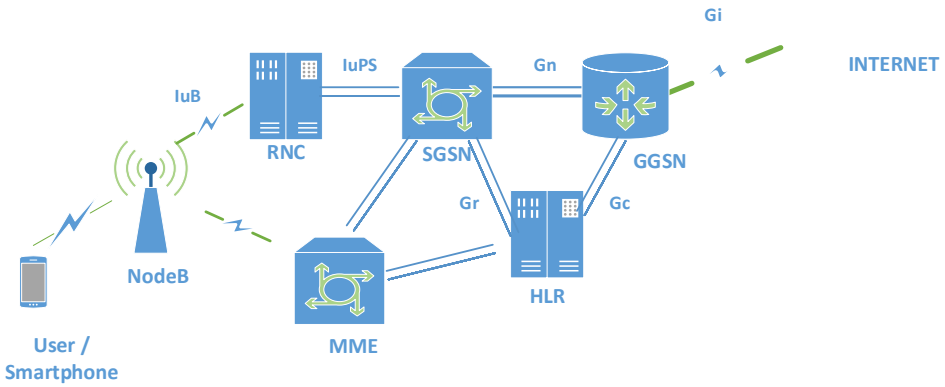


Figura 5.11: Arkitektura e thjeshtuar e UMTS/LTE për të dhënat e UE

5.5.1 Metodologjia 1: Përdorimi i UE si modem në PC

Topologjia e matjeve jepet si në figurën 5.12 ku një kompjuter ***Kk*** i lidhur me kabëll në rrjetin publik të Internetit dërgon paketa te një përdorues pa tel ***M,*** i lidhur në rrjetin nën testim nëpërmjet një pajisjeje modem USB 3G. Të dy përdoruesit mund të egzistojnë ose konfigurohen në të njëjtën makinë fizike apo PC (me bazë sistem operimi Windows apo Linux), në mënyrë që të kemi të njëjtën orë (timestamp) në matjet përkatëse. Të gjitha paketat mbërrijnë në të dy ndërfaqet (kabllore dhe ajrore) kapen dhe ruhen (koha) me përdorimin e programit *tcpdump* dhe vonesat

njëdrejtimëshe janë llogaritur si ndryshime në kohët e mbërritjes. Saktësia e orës së PC (kompjuter personal) duhet të jetë e rendit të 0.1- 0.2 ms që mbulon qëllimin e analizës sepse vonesat në 3G e 4G janë të rendit disa dhjetra ms.

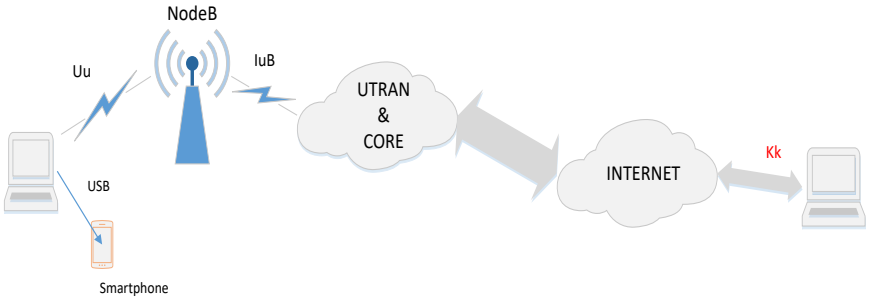


Figura 5.12: Metodologji e matjes duke përdorur lidhjen 3G si një modem në një PC

Në shumicën e testimeve përfshihen transmetime paketash vetëm në DL – lidhjen zbritëse, ku përdoruesi mobile *M* i është bllokuar dërgimi i paketave në UL – lidhjen ngjitëse në mënyrë që të minimizohen konfliktet me kontrollin me lidhjen zbritëse DL. Përdoren kërkesa-paketa *echo* ICMP me një gjatësi totale pakete prej L =xxx Byte (psh 100 Byte). Gjatë eksperimentit mund të dërgojmë nga përdoruesi kabllor *Kk* një seri paketash (të shënjuara si *k*) gjatë një intervali kohor fiks τ drejt përdoruesit mobile *M*. Fillimisht, ne dërgojmë të dhëna me shpejtësi të lartë (pra τ *e ulët*) dhe përsërim eksperimentin duke rritur vlerat e τ.

Përcaktojmë si $N_K(t)$ dhe $N_M(t)$ respektivisht proceset e numërimit të Byteve të dërguara nga *Kk* dhe të marra nga *M* në intervalin kohor [0: t]. Në figurën më poshtë mund të nxirret grafiku i $N_K(t)$ dhe $N_M(t)$ për një eksperiment. Grafiku tregon dhe përmban në vetvete një grup informacionesh rreth disa parametrave radio të rrjetit. Duke shënuar respektivisht me *B_cch* dhe *B_dch* kapacitetin e ofruar të një përdoruesi i vetëm MS në kanalet CCH dhe DCH. Gjatë testimit dhe analizës në figurën e gjatësisë së segmentit, *oa* paraqet vonesën në DL në kanalin CCH, e shënuar si *Vcch*. Të njëjtat paketa janë transferuar gjithashtu nëpërmjet CCH, por meqenëse shpejtësia e dërgimit është më e lartë sesa kapaciteti i kanalit CCH, psh L = τ >B_cch, ato kalojnë në pritje dhe vonesa rritet. Vlera aktuale e *Bcch* është e njëjtë me pjerrësinë e segmentit *ab* si te figura më poshtë. Pas rreth 1 sekonde (pika b) rrjeti, e specifikisht kontrolluesi i rrjetit radio apo RNC, vendos t'i caktojë një kanal DCH përdoruesit UE.

Proçedura e caktimit të kanalit DCH kërkon një kohë të shpejt kompletimi të shënuar si *Ddch:on* gjatë së cilës dërgimi i paketave në linkun radio është ndërprerë. Kjo vlerë e *Ddch:on* një tregues performance dhe vlera aktuale e tij mund të vlerësohet direkt nëpërmjet matjes së segmentit *bc*. Gjatësia e *bf* riparaqet shumën maksimale të Byte-ve të bufferuar përpara se të kalohet në kanalin DCH. Në pikën *c*, kanali DCH është i mundur të vendoset dhe paketat e bufferuara transmetohen me shpejtësinë e *Bdch*, vlera e të cilës mund të shihet nga vlerësimi i segmentit *cd*.

Në mënyrë të qartë mund të thuhet se *B_dch* >> *B_cch*. Në pikën d, bufferi është përsëri bosh. Pas kësaj pike, distanca horizontale midis 2 kurbave riparaqet vonesën në DL në kanalin DCH të shënuar si Vdch.

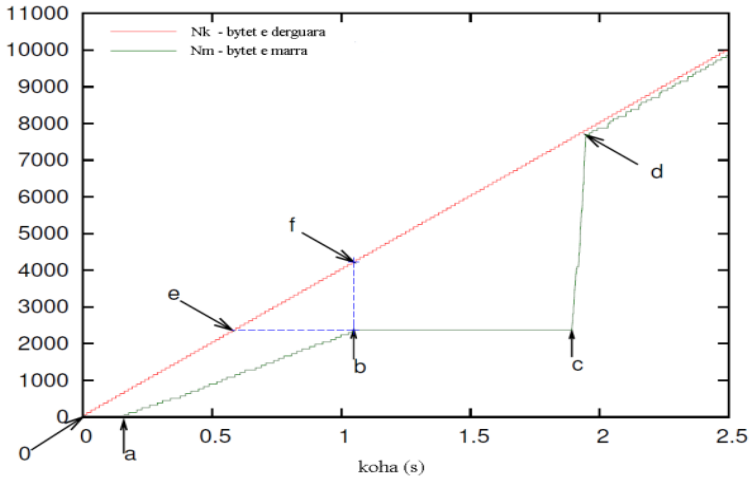


Figura 5.13: Rezultati i matjes me metodologjinë 1

Mund të ripërsëriten testet për vlera të ndryshme të paketave të transmetuara me intervale kohore $\boldsymbol{\tau}$ dhe gjatësi L të ndryshme. Kur shpejtësia e dërgimit të paketave bie poshtë një vlere të caktuar, kalimi në kanalin DCH nuk vendoset nga RNC dhe UE mbetet në CCH. Në këtë mënyrë, me anë të vlerave të ndryshme të $\boldsymbol{\tau}$ dhe L mund të arrihet të kuptohet algoritmi i caktimit të kanalit DCH. Lëshimi i kanalit DCH bazohet në një kohëzues bosh (timeout), që mund të rivihet pas transmetimit të çdo paketë (të marrë ose të dërguar). Vlerën e këtij parametri e shënojmë si *Tdch:off* apo T_1 dhe kjo është e modifikueshme si parametër vetëm nga operatori i rrjetit UMTS. Për të zbuluar vlerën e këtij parametri ne mund të testojmë duke detyruar UE të kërkojë një kanal DCH, duke dërguar një burst të madh fillestar të dhënash. Kjo është e njëjta fushë sikurse në pika d në figurën 5.13. Ne, më pas mund të dërgojmë një sekuencë paketash me rritje të ngadaltë të mbërritjes në kohë (koha τ e përmendur më sipër) apo IDT. Vlera e t_k e IDT jepet si:

$$\tau k = \tau k_i 1 + ¢\tau \quad (5.10)$$

ku $¢\tau$ është një rritje konstante apo fikse. Pas njëfarë kohe, UE kalon pas në kanalin CCH. Vlera e fundit e IDT mund të përdoret si vlera e *Tdch:off* . Tranzitimi i CCH mund të identifikohet dhe në mënyra të tjera sikurse do ta shohim në metodologjinë 2, ku disa pajisje UE apo modema USB raportojnë hollësisht ndryshimet në kanal.

Realizimi praktik kërkon që pikë së pari që UE apo MS të jetë i aksesueshëm nëpërmjet internetit dhe paketat e dërguara të jenë aktualisht të marra nga UE, dhe jo të filtruara (për shembull nga një firewall) në rrjetin UMTS. Ne mund të bëjmë një skanim portash me *nmap* dhe *hping* te të gjithë portat TCP në një telefon UE UMTS apo LTE. Shpesh shihet se pothuajse të gjitha portat janë të hapura. Pas kësaj ne kërkojmë të shikojmë nëse është mundur të "zgjojmë" telefonin nëpërmjet ping-eve. Eshtë gjendur që shumica e tyre (portave) janë të hapura/lejuara, dhe UE gjithashtu i përgjigjet atyre. Mund të përdoren aplikime të ndryshëm që lejojnë të dërgojmë trafik te një MS dhe ta zgjojmë atë në një gjendje konsumi me energji të lartë.

Burimi i trafikut (inicuesi), një kompjuter me sistem operativ të përshtatshëm për të mbështetur lidhjen në internet, programet aplikative si Wireshark, tshark apo të ngjashme për sistemin OS Android (për verifikimin e dërgimit të paketave dhe kohëve përkatëse) dhe toolse / mjetet për

dërgimin e paketave testuese si *Ping*, *nmap hping e UDPsender*: përdoren për të dërguar apo iniciuar paketa. *Pritësi i trafikut* (dëgjuesi) përfshin; një smartphone apo një modem 2G/3G i lidhur në një PC me sistem operativ të përshtatshëm për të mbështetur lidhjen në internet, programet aplikative si Wireshark (për verifikimin e dërgimit të paketave dhe kohëve përkatëse), aplikimin firewall për të filtruar trafikun e padëshiruar (aplikime si Ghostwall apo DroidWall) dhe aplikime te smartphonëve në PC (për lidhjen si Nokia PC Suite apo Android SDK, Kies etj).

5.5.2 Metodologjia 2: Përdorimi i aplikimeve matëse dhe pajisjeve fizike në UE

Për realizimin e metodologjisë 2 përdoren pajisje UE "të zgjuar" apo smartphone, të cilët kanë të implementuar në softin e tyre një modul/aplikim SW për matje rrjeti radio. Në testimet që në propozojmë në këtë metodologji janë përdorimi i telefonave Nokia (si psh N95,) të cilët kanë si sistem operimi Symbian OS (megjithatë mund të përdoren dhe telefona me sistem operimi Android apo iOS). Në këto pajisje MS mund të instalohet dhe të përdoret moduli SW NEP (ka dhe SW më profesional si Nemo apo dhe aplikime me bazë Android OS si Powertutor e Trepn) i zhvilluar nga Nokia, që është modul SW për matje të ndryshme në lidhje me energjinë apo kohëzuesit. Gjithashtu, u jep mundësi analistëve të rrjetit apo zhvilluesve të aplikimeve SW informacion mbi:

- Energjinë e konsumuar gjatë një aktiviteti.
- Shpejtësinë e të dhënave.
- Fortësinë e sinjalit në DL (nga BTS drejt UE).
- Kohëzuesit e rrjetit radio 3G, si T_1 e T_2.

Ky aplikim shërben për matje pa përdorur pajisje HW shtesë. Megjithatë, testuesit apo analizuesit e ndryshëm për të kontrolluar dhe korrigjuar të dhënat nga matjet mund të përdorin paralelisht dhe një sërë pajisjesh shtesë sikurse në figurën 5.14. Përfshihet një multimetër profesional i cili mund të lidhet me një PC i cili ka software-in përkatës (si psh Agilent 66319D), i cili është i dizenjuar për të matur fuqinë e baterive të pajisjeve portable si atyre të UE. Në këtë mënyrë mund të krahasohen rezultatet e matura në UE dhe paralelisht në pajisjen HW. Të dy pajisjet lidhen te i njëjti PC në mënyrë që të ruhet e njëjta kohë në testime. Pajisja UE lidhet nëpërmjet një USB-je dhe moduli NEP lihet të punojë në background / sfond. Në anën tjetër pajisja HW apo multimetri lidhet me daljet e baterisë së pajisjes MS duke matur konsumin e energjisë gjatë aktiviteteve të ndryshme që kryen përdoruesi UE.

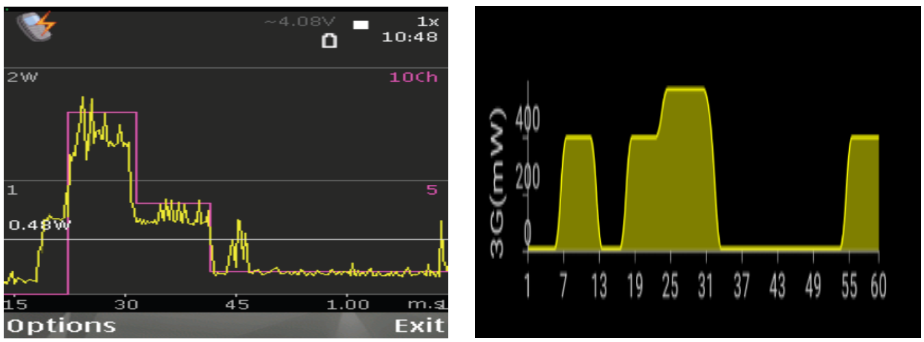


Figura 5.14. Metodologjia II e matjes me Vm dhe aplikimin NEP [1] e PowerTutor

Nëpërmjet kësaj metodologjie arrihet të analizohen energjitë e kërkuara në kanale të ndryshëm si: 1) *energjia e kërcimit (ramp)* për të kaluar nga një gjendje e ulët në një gjendje të lartë, 2) *energjia e transmetimit,* 3) *energjia e bishtit (tail)* apo energjia e konsumuar në MS duke qenë në gjendje energjie të lartë (DCH) pas kompletimit të një transferimi të dhënash.

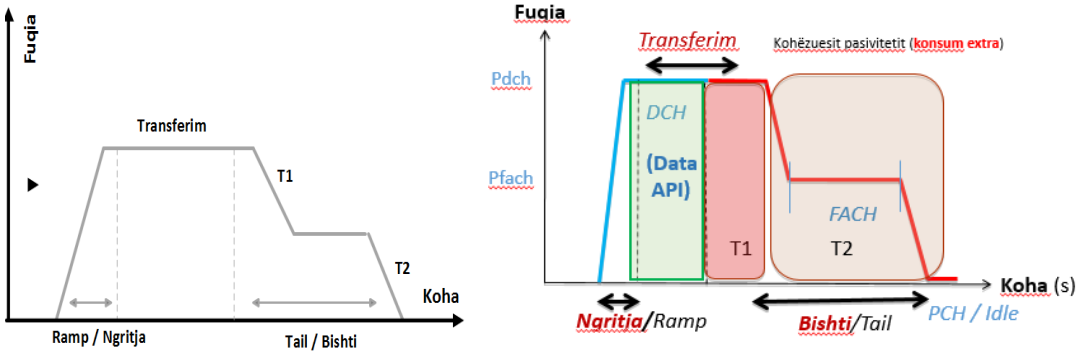


Figura 5.15: Fazat energjitike në një komunikim 3G / UMTS

Në krahasim me metodologjinë e parë në këtë metodë testet zhvillohen duke dërguar e marrë të dhëna nga UE drejt një serveri fundor me përmasa të variueshme të paketave apo transferimeve M_{dt} (1 deri 1000 KB) me intervale pushimi T_p (1 deri në 30 sekonda) midis transferimeve të suksesshme. Ne masim energjinë e konsumuar gjatë çdo transferimi nëpërmjet NEP-it dhe multimetrit, ku NEP-i punon në background ndërsa të dhënat transferohen në DB e tij. Për çdo konfigurim të (T_p, M_{dt}), testi i matjeve zhvillohet në këtë formë: UE inicion një *downloadim* M_{dt} KB duke iniciuar një kërkesë *Http* nga një server në distancë. Pasi shkarkimi ka përfunduar, UE pret për T_p sekonda dhe inicion një kërkesë tjetër. Procesi duhet të përsëritet për të bërë një mesatarizim të vlerave të matura. Midis testimeve të transferimit të të dhënave me intervale të ndryshme T_p, UE qëndron në idle për 60 sek.

I njëjti eksperiment mund të kryhet *për të dërguar* të dhëna në një server në distancë. Ne mund të nxjerrim të dhënat nga NEP, të ruajtura në një format excel, duke përdorur kohën e regjistruar nga NEP për të treguar fillimin dhe mbarimin e transferimit të të dhënave si dhe kohët e përshkruara më sipër të cilat në lidhje dhe me figurën 5.15 tregojnë tranzicionet midis burimeve radio, apo dhe shpejtësitë e kërkuara për tranzitim (në paralelizëm me metodologjinë e parë). Si përfundim nga matje të kryera nga NEP si dhe nga multimetri rezulton një përputhje shumë e lartë në vlerat e matura në lidhje me:

Koha e mosaktivizimit	**Operator 1**	**Operator 2**
DCH --> FACH	5 sek	6 sek
FACH --> IDLE	12 sek	4 sek
Koha e promovimit	**Operator 1**	**Operator 2**
QETE (IDLE) --> FACH	N/A	0.6 sek
QETE (IDLE) --> DCH	5 sek	N/A
FACH --> DCH	1.5 sek	1.3 sek
Kufiri i buferit RLC	**Operator 1**	**Operator 2**
FACH --> DCH (UL)	543 ± 25 B	151 ± 14 B
FACH --> DCH (DL)	475 ± 23 B	119 ± 17B
Fuqia e gjendjes radio	**Operator 1**	**Operator 2**

DCH / FACH / QETE	800 / 460 / 0 mW	600 / 400 / 0 mW
Fuqia e promovimit radio	**Operator 1**	**Operator 2**
QETE (IDLE) --> FACH	N /A	410 mW
QETE (IDLE) --> DCH	550 mW	N /A
FACH --> DCH	700 mW	480 mW

Tabela 5.4: Shembuj parametrash të RRC e RLC të matshme

Për fillimin e testimeve është e nevojshme të hiqet bateria e UE dhe të vendoset midis baterisë dhe telefonit në seri një rezistor me vlerë shumë të ulët psh 0.02 Ohm, dhe një matës i tensionit (në paralel me rezistencën) në këtë rezistencë lidhet dhe një matës i dytë që lidhet direkt në baterinë e telefonit. Diagrama e qarkut jepet si në figurën 5.16. Matësi i tensionit të baterisë mat tensionin e çastit duke përdorur ligjin e Kirkovit dhe prej nga ku nxjerrim rrymën e çastit të kërkuar nga telefoni UE e cila mund të llogaritet me formulën e mëposhtme:

$$I = \frac{xV - U}{R} \quad (5.11)$$

$$I = \frac{4.1V - U}{R}$$

Ku xV tensioni në dalje të baterisë, **U** është tensioni i çastit i matur nga matësi dhe **R** resistenca prej ~0.02 Ohm (ideale nëse do sigurohej për testime).

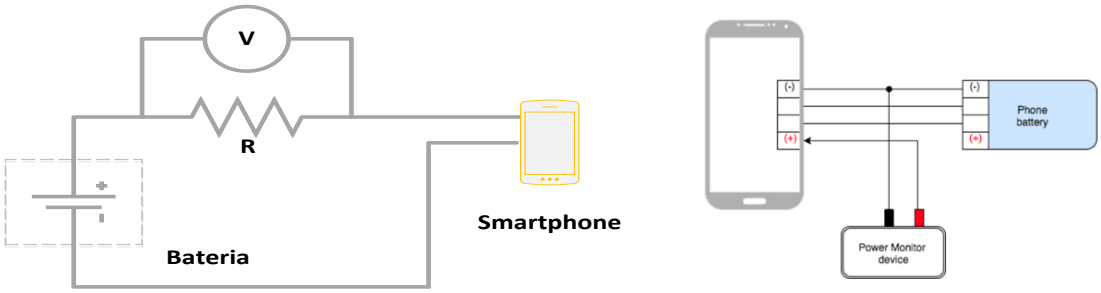


Figura 5.16: Skema e lidhjes për matjen e tensionit/rrymës së çastit në një UE

5.5.3 Metodologjia 3: Përdorimi i filave loger apo Wireshark në UE

Mund të përdoren për trace dhe tools-e *Netlog* në pajisjet Windows mobile dhe aplikime si *tcpdump, tshark* në telefonat Android. Trace-t mund të ruhen lokalisht dhe të dërgohen apo mblidhen periodikisht në intervale kohore për procesime të mëtejshme. Në këtë lloj forme nuk kemi interferencë në lidhje me përdorimin e pajisjeve apo aplikimeve të jashtme të cilat mund të kërkojnë një shtim energjie krahasuar me atë të kërkimit nga aplikimet nën testim. Në përgjithësi në telefonat Android kërkohet memorje e jashtme SD për të ruajtur këto fila matjesh.
Për të eleminuar trafikun jo të duhur data, për studim mund të përdoren aplikime të cilat mund të filtrojnë trafikun e aplikimeve të tjera si p.sh Ghostwall e DroidWall.

Në lidhje me testimet mund të injektohet trafik me karakteristika të ndryshme në rrjet, merret trafik në kartën tjetër të rrjetit dhe ruhen kohët e dërgimit (timestamp), vonesa njëdrejtimshe dhe IDT (Inter-Departure Time) për çdo paketë. *IDT-ja është vonesa midis dërgimit të paketave të*

njëpasnjëshme. Duke rritur IDTn në mënyrë të ngadaltë, mund të zbulohet dhe mbarimi i kohëzuesit të pasivitetit.

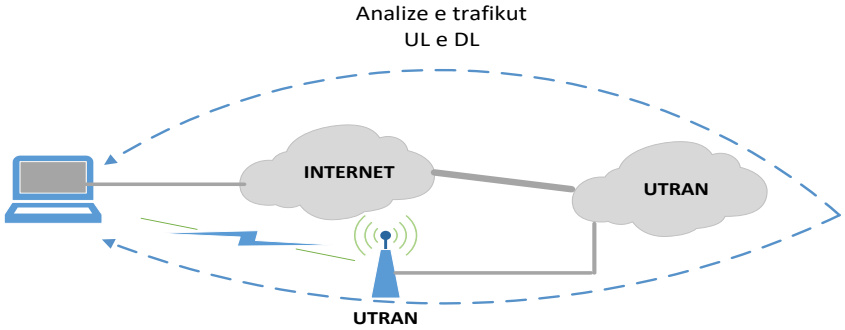


Figura 5.17: Metodologjia 3 duke përdorur fila "logger"

Zakonisht, informacioni i baterisë në telefonat Android OS është i disponueshëm përmes ruajtjes së të dhënave ne skedarët në: */ sys / class / power_supply / battery*. Skedari dhe përmbajtja e tij varen nga modeli i pajisjes celulare. Në shumë pajisje (p.sh. Samsung e ASUS) këto skedarë përmbajnë vetëm informacionin aktual të tensionit. Keto fila mund të nxirren nga ky skedar duke përdorur komandatat e toolsit ADB's si:

adb pull ... /folder/

adb push ../folder/

5.6 Mbledhja e të dhënave

Këtu ne fokusohemi në kapjen e të dhënave nga aplikimi dhe metoda e përdorur për kapje paketash. Paketat e gjurmuara/vëzhguara janë kapur nga një telefon smartphone që zhvillon teste automatike duke përdorur software të kapjes së paketave, i instaluar në telefon (si tshark). Aplikime bazë UE mund të përdoren për të përcaktuar konsumin e energjisë të aplikimit dhe shkaqet e bishtave të energjisë duke analizuar log-e automatike të mbledhura nga sistemi apo kerneli i tij.
Çdo tranzitim i gjendjeve trigeron në RNC do të shkëmbejë mesazhe sinjalizimi me UE në rrafshin e planit të kontrollit (shark, tshark apo wireshark). Respektivisht, nga disa matje për një tip aplikimi rezulton një numër specifik mesazhesh sinjalizimi si:

Tranzitim i gjendjeve RRC	**Nr. i mszh të sinjalizimit**
IDLE → DCH	23
DCH → FACH	4
FACH → DCH	10
DCH → IDLE	8
FACH → IDLE	6

Tabela 5.5: Numri i mszh-ve të sinjalizimit të gjeneruar për çdo tip tranzitimi gjendjeje RRC

Sikurse shihet kalimet nga IDLE në DCH dhe FACH në DCH dominojnë më së shumti me mesazhe sinjalizimi.

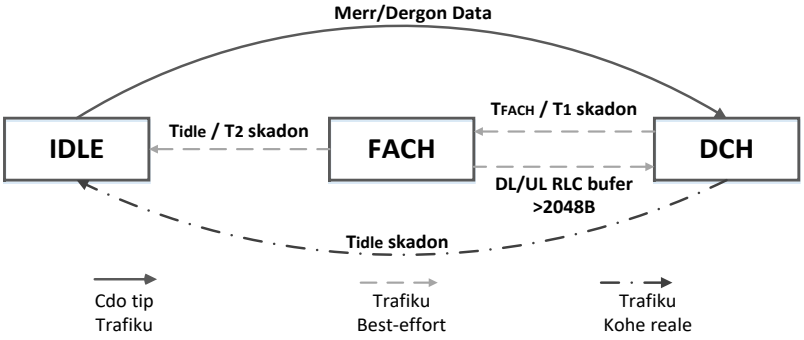


Figura 5.18: Fluksi dhe kushtet e tranzitimit të trafikut UL/DL

a. Shpërndarja e mesazheve të sinjalizimit

Në tabelën në vijim jepet një përmbledhje e tranzitimit të gjendjeve në RRC dhe kontributi në numër i mesazheve të sinjalizimit në një rast komunikimi. Vërehet se për tranzitimet:

- kalimi IDLE → DCH kontribuon me > 40% të mesazheve të sinjalizimit.
- kalimi DCH → IDLE dhe FACH → IDLE së bashku kontribuojnë në rreth 18% të mesazheve gjithsej.

Tranzitim i gjendjeve RRC	Perqindja e mszh sinjalizimit
IDLE → DCH	42.48%
DCH → FACH	14.12%
FACH → DCH	25.03%
DCH → IDLE	11.86%
FACH → IDLE	6.15%

Tabela 5.6 Përqindja e mszh të sinjalizimit si kontribut i ndryshimit të gjendjeve

b. Paketat Uplink (UL) përkundër Downlink (DL)

Në këtë nënseksion është e rëndësishme të ceken detaje të cilat tregojnë dhe fluksin dhe drejtimin e të dhënave në komunikime data për disa teste të kryera nga një grup studiuesish në këtë fushë. Sikurse mund të shihet:

- Shumica (>80%) e tranzitimeve janë detyruar nga UL. Pra update që mund të vijnë nga rrjeti / apo serverat në shërbim.
- Kalimi Idle → DCH kontribuon në shumicën e tranzitimeve dhe mesazheve të sinjalizmit si për drejtimin UL dhe atë DL. Kjo parametrizohet nga vetë operatorët.

c. *Trafiku bazë TCP përkundër UDP*

Gjithashtu në këtë nënseksion është e rëndësishme të ceken detaje të cilat tregojnë tipin e trafikut për protokoll të përdorur:

- Shumica e paketave të trafikut që trigerojnë ndryshim gjendjesh janë si shkak i **TCP në drejtimin UL**. Pra ma pak fjalë i kërkesave të përdoruesit duke përjashtuar shkarkimet;
- Trafiku UDP trigeron vetëm një pjesë të vogël të proporcionit të tranzitimeve (me rreth 13%).

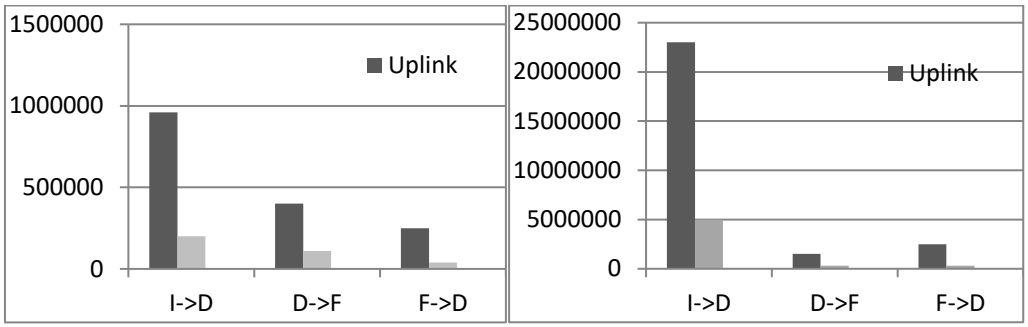


Figura 5.19: Numri i tranzitimit të gjendjeve dhe nr i paketave në UL dhe DL

Ky dallim shërben për të theksuar natyrën e komunikimit apo protokolleve të përdorura nga aplikimet si dhe ndikimin që kanë në komunikim. Kjo është e nevojshme pasi egziston një dallim thelbësor midis UDP e TCP në dërgimin e të dhënave dhe pritjen e përgjigjeve – njohjeve ACK, etj. Kjo në mënyrë të drejtpërdrejtë ka lidhje dhe me zgjidhjen midleware që në mendojmë.

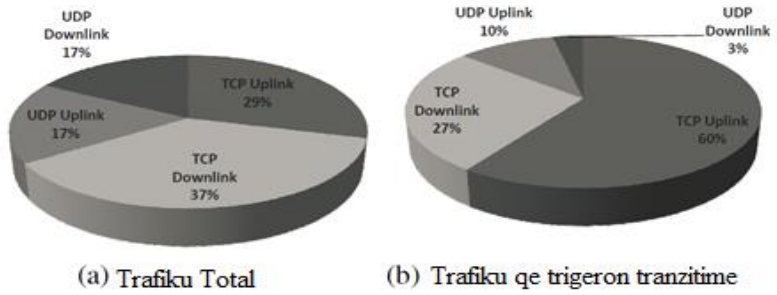


Figura 5.20: Shembull tipi i trafikut dhe tranzitimet për lloj trafiku

d. Analiza e flamurit TCP

Praktikisht protokolli TCP shkëmben shumë mesazhe në rrjet gjatë një komunikimi c'ka e bën UE'n të qëndrojë i lidhur apo i "i sinkronizuar" me rrjetin, duke e mbajtur atë gjatë në një prej kanaleve të komunikimit. Devijimi prej këtij funksioni (si psh vonesat) do te sillte gabime, ritransmetime etj në komunikim.

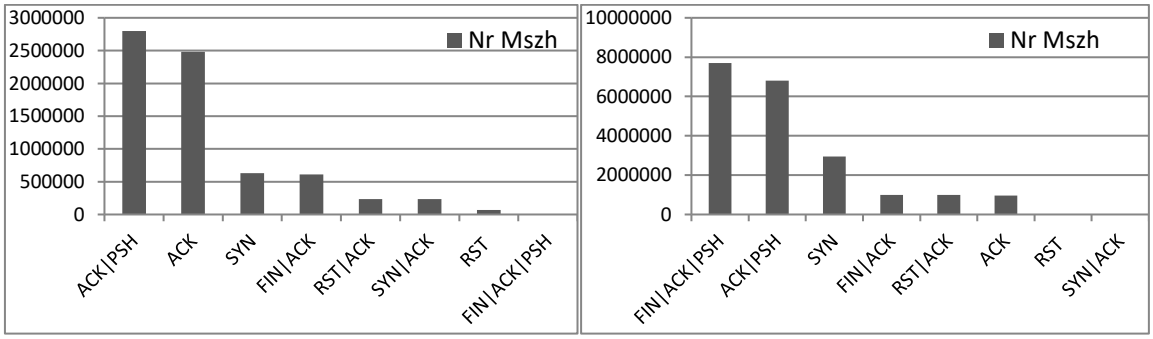


Figura 5.21: Numri i tipit të mesazheve të sinjalizimit në: *a.*DL dhe *b.*UL

- Dallohen 8 tipe kryesore të paketave TCP në çdo drejtim komunikimi.
- ***Paketat UL me SYN, FIN, ose flamujt RST*** *kontribojnë një pjesë të rëndësishme/me peshë në mesazhet e shkembyera.*

- Shumica e mesazheve janë si shkak i fluksit nga ***Idle → DCH*** në përputhje dhe me atë ç'ka përmendëm dhe më lart.

*Ndërkohë me përdorimin e protokollit **UDP** një pjesë e këtyre mesazheve (në përbërje ACK) nuk kërkohet dhe kjo lehtëson komunikimin (në lidhje me një vonesë të lehtë). Aplikime që përdorin këtë protokoll dhe që shërbejnë në background janë në interesin tonë gjatë këtij studimi.*

5.7 Matje dhe analiza të konsumimit të energjisë për Smartphone UE

Në këtë seksion ne do të paraqesim një analizë të detajuar të konsumimit të fuqisë për nga një celular smart të kohëve të fundit, si modeli Nokia Lumia 625 në Windows OS dhe Samsung S3/4 GT I9100 në Android. Ne matëm dhe diskutuam rëndësinë e harxhimit të fuqisë nëpërmjet komponentëve, aplikacioneve të ndryshme dhe identifikuam zonat më premtuese për tu përqendruar në përmirësime të mëtejshme të menaxhimit të fuqisë. Ne duam të paraqesim ndërprerjen e energjisë në etapa si edhe në një numër skenarësh realistë përdorimi.
Aftësia për të gjeneruar një model të shkallëzuar të konsumit të fuqisë me pajisje specifike është kësisoj e rëndësishme për të kuptuar, dizenjuar dhe zbatuar një software më të mirë aplikacioni celular.
Një infrastrukturë e duhur e përllogaritjes së energjisë do të ndihmojë si zhvilluesit e aplikacionit edhe përdoruesit e Smartphone-eve të zgjasin jetën e baterisë për pajisjet e tyre dhe të marrin vendime se ku të shpenzojnë fuqinë e mbetur të pajisjes (në matje në kohë reale).
Smartphon-ët modern përdorin SoC heterogjene multi-bërthamore e cila përfshin CPUn, GPU, DSPs dhe përshpejtuesit specifikë të aplikacioneve të ndryshme. Ajo ofron mundësi për të realizuar aplikacione intensive vajtje-ardhje në pajisje të lëvizshme me burime të kufizuara në lidhje me fuqinë (nga bateria) duke i caktuar çdo detyre ose veprimtarie të përdoruesit një pjesë të funksioneve të bërthamës / DSP-ve. Funksionaliteti i pasur me Hw e APIs rrit presionin mbi jetëgjatësinë e baterisë, dhe thellon nevojën për menaxhimin e energjisë efektive [1].
Në fakt, celularët e sotëm Smartphone shfaqin dallime të rëndësishme përsa i përket nënshkrimeve të konsumit të energjisë në varësi të prodhuesit, sistemit operativ dhe faktorëve të tjerë kontekstual të tilla si mbulimi i rrjetit. Të kuptuarit se si energjia konsumohet nga komponentet hardware është thelbësore për dizenjimin e sistemeve të vetëdijshme të energjisë.
Testet tona janë bërë duke qenë statike (jo në lëvizje) dhe në disa teste ne tregojmë nivelin sinjal RSSI, që duket të jetë një nga faktorët kryesorë që duhet të konsiderohet. Ka një sërë testimesh dhe matjesh të kryera nga autorë të ndryshëm në telefonat celularë realë (pothuajse të gjitha punimet përfshijnë testime) dhe në pajisje testuese me arkitekturë të hapur si *Openmoko Neo Freerunner*. Nokia Lumia 625 është një Windows OS celular 8.0 modeli (versioni ynë SW është 8.0.10517.0 për periudhën e testeve) dhe ka dy pamje kryesore si në figurë:

Figura 5.22: Menuja kryesore dhe kalimi te menuja e aplikimeve (Nokia e Android)

Për të llogaritur fuqinë e konsumuar nga çdo komponent, duhet të përcaktohet si tensioni dhe rryma elektrike. Për të matur rrymën elektrike, ne kemi futur një rezistencë të vogël midis baterisë e pjesës së furnizimit me energji të komponentit përkatës apo UEs si në figurën më poshtë. Kjo respektivisht me metodologjinë 2 të shprehur më parë te metodologjitë e matjeve.

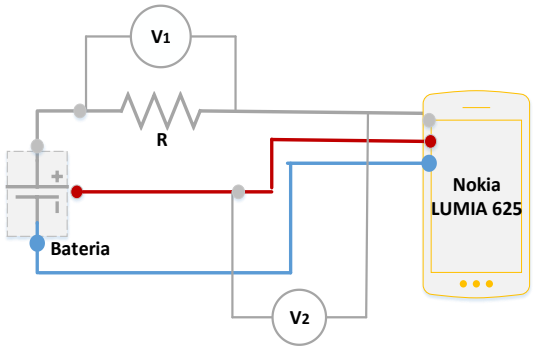


Figura 5.23: Skema e matjeve të rënieve të tensionit në UE

Rënia e tensionit të pikut në rezistencën shtesë nuk i kalon më shumë se 2-3% të tensionit të furnizimit dhe për këtë arsye ajo paraqet një ndryshim të vogël të pranueshëm. Me një rezistencë të njohur dhe rënie tensioni të matur, rryma elektrike mund të përcaktohet lehtë me ligjin e Ohm-it. Për lehtësinë tonë, ne kemi paraqitur për llogaritje vetëm rënien e tensionit mbi rezistencë R = 0.22 Ohm, rrymën elektrike dhe fuqinë e konsumuar.
Prandaj, janë krijuar disa modele statistikore të cilat ndajnë konsumin e energjisë në nënsisteme, por që nuk janë konsideruar në rastin tonë për momentin. Një formulë e përgjithshme llogaritëse do të ishte formula 5.12, duke përmbledhur impaktin e çdo elementi apo ndërfaqeje:

Fuqia UE = CPU&Ram+Display+Graphics+Network+GPS+Audio+Mic+....+WiFi (5.12)

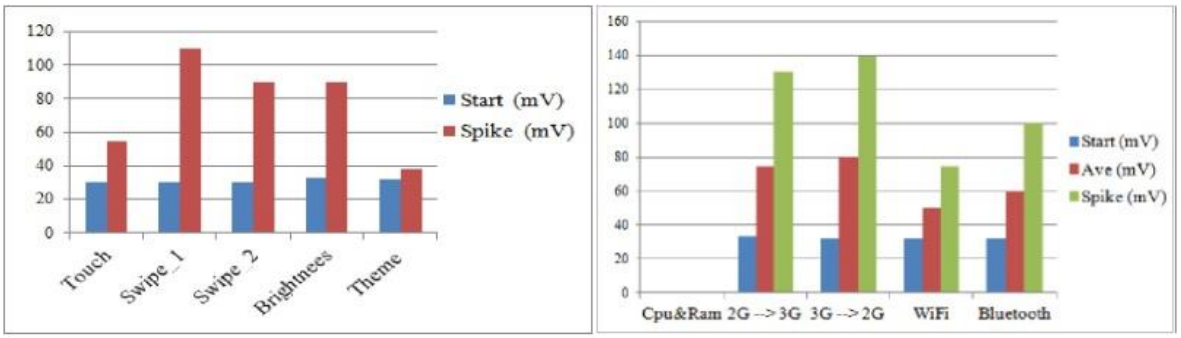
Ne ngritëm dy lloje kryesore krahasimesh/etapash të mbajtura nën të njëjtën logjikë si për autorët në [1]. Së pari, një seri e *mikro-vlerësimeve* të dizenjuara për të karakterizuar në mënyrë të pavarur komponentët e sistemit, veçanërisht konsumimi normal i përdorimi të energjisë së tyre (rëniet e tensionit). Së dyti, ne krijuam një sërë *makro-vlerësimesh* bazuar në skenarët realë të përdorimit. Pak aplikacione software të telefonit janë përdorur për të ndihmuar në drejtimin e testeve shtesë. Për shumicën e provave, ne kryem 10 përsëritje.

Matja e konsumimit të fuqisë së telefonit është matur duke përdorur një rezistencë precizioni të rendit 0,2 Ohm të lidhur në seri me baterinë e terminalit dhe e lidhur sikurse në figurën 5.19 (metodologji e përdorur shumë gjerë nga studiues të ndryshëm në këtë fushë). Ne përdorëm si pajisje matëse fizike një *Digital meter GWInstek – GDM-8246 sampling board* (0.02% DCV saktësi) dhe *DT9205A digital multimeter* për të matur direkt tensionin në baterinë e telefonit dhe gjithashtu rënien e tensionit në rezistencën e vendosur e cila është e lidhur me pajisjet nëpërmjet një kablli 2- fijësh sikurse në figurë.
Matjet tona në smartphone-in Nokia tentojnë të nxjerrin në dukje fuqinë e konsumuar direkt (rënien e tensionit) te një grup i komponenetëve të UE dhe gjithmonë duke përfshirë në çdo matje harxhimin e sjellë nga CPU dhe RAM. Ndërsa me smartphone-in Android qellimi i matjeve është të matet kryesisht energjia e konsumuar te ndërfaqja e rrjetit 3G. Duhet thënë së shumë shërbime të reja në analizimin e energjisë së konsumuar/matur zgjasin një fraksion të sekondës. P.sh: një skanim i rrjetit wireless zgjat vetëm 500 ms, ndërsa transmetimi i një pakete të vetme mund të marrë disa milisekonda (ms) te transmetohet . Kyçja e ekranit të telefonit nga gjendje aktive në gjendje joaktive ul në mënyrë të ndjeshme fuqinë e konsumuar në pajisje dhe jep mundësinë që në shumë raste të masim energjinë e një komponenti specifik ose grupi komponentësh.

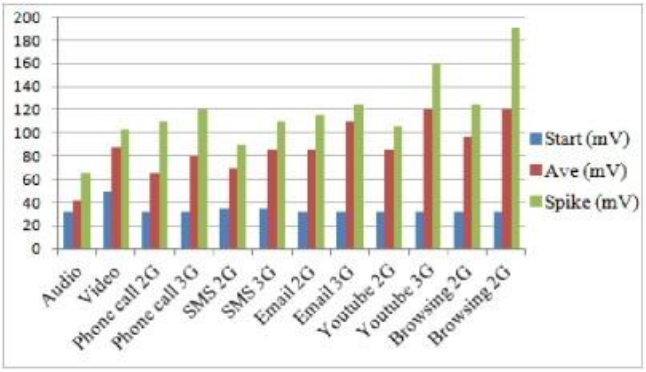
5.7.1 Rezultatet përmbledhëse

Atëherë si konkluzion ne këtu mund të themi se jo lineariteti i nënsistemeve është një çështje e njohur dhe dihet që ato shkaktojnë një konsum fuqie në një interval të dhënë të kohës, e cila varet nga gjendja në të cilën pajisja UE operon. Psh. ngjyra të ndryshme në piksela dhe gjendjet e ndriçimit kanë një efekt të madh në *shumën e energjisë* së konsumuar nga ekrani i telefonit. Rezultatet tona nga matjet me Nokia tregojnë që shumica e energjisë së konsumuar (rënies së tensionit) mund t'i atribuohet ndërfaqes së rrjetit dhe ekranit, përfshirë panelin IPS LCD, prekjen, zhvendosjen dhe parametrat për ndriçimin, ngjyrat etj. Në të gjitha testimet përveç atyre 3G/4G e 2G, ndriçimi i ekranit është një nga konsumuesit më të mëdhenj.
Megjithatë kjo varet në mënyrë direkte nga preferencat e përdoruesit. Të luajturit e një videoje apo muzike influencohen më shumë nga faktorë si kohëzgjatja, tipi MP3 apo MP4, pra kodeku i përdorur, shpejtësia e kampionimit, ndriçimi si dhe volumi zërit etj. Ndikimi i Audios në teste tregoi një ndikim të vogël e statik. Gjatë testimeve të rrjetave për thirrjet, rezultoi se rrjeti GSM,3G e LTE konsumon më pak energji kur DTX dhe DRX janë të aktivizuara në nyjet e rrjetit.
Ndryshimi i gjendjes së ekranit gjatë një thirrjeje mund të rezultojë në një ruajtës energjie të mirë. Navigimi në internet, luajtja e videove në Youtube dhe shkarkimi apo ngarkimi i filave nga telefoni janë më të përdorshmet nga përdoruesit (Email, SN, Viber, WhatsApp etj), por dhe ato me konsumin më të lartë të energjisë. Kur pajisja UE është në gjendje IDLE konsumi i energjisë është i ulët dhe kjo i atribuohet OS e cila i vendos në gjendje gjumi komponentët dhe nënkomponentët e UE.
Të pasurit e më shumë aplikimeve në menun kryesore dhe me kushtin e lejimit të updateve në background/sfond mund të rrisë në mënyrë të ndjeshme rënien e shpejt të baterisë. Figurat më poshtë japin një përmbledhje të rezultatetve nga testimet e kryera.
Ekrani dhe nderfaqja e rrjetit (komunikimi për të dhënat) shikohen si dy dominuesit kryesorë të konsumit të shpejt të energjisë. Në interesin tonë është ndërfaqja e rrjetit e cila është drejtëpërdrejtë

e lidhur me mënyrën sesi aplikimet ndërveprojnë me rrjetin & serverat duke ndryshuar gjendjen e ndërfaqes së rrjetit dhe për rrjedhojë rritje të konsumit të energjisë.



a) Rastet Bazë b) Micro benchmarks



c) Macro benchmarks

Figura 5.24_1: Rezultate matjesh në Nokia Lumia 625

Një nga limitimet në këto matje ka qenë fakti i mosdhënies në detaje, me saktësi të lartë të vlerave të matura të rënies së tensionit për elemente specifike, pasi kjo nuk mundësohet me këtë model DuT, për shkak të limitimeve SW Nokia 625 të OS Windows Mobile.

Sic thamë janë kryer dhe një seri matjesh të tjera duke duke përdorur një telefon UE Samsung S3/4 GT-I9100 me sw OS Android e kartë SIM nga operatori shqiptar *066 6110779* i pozicionuar në 3G për të siguruar akses të plotë të kapaciteteve të rrjetit në dispozicion dhe për kushte të pranueshme në sinjalin e marrjes apo RSSI. Përveç nëse specifikohet ndryshe, të gjitha matjet janë kryer në të njëjtin vend / pozicion fiks në një zonë ku forca e marrjes së sinjalit RSSI nuk ndryshon apo varion shpejt e në mënyrë të konsiderueshme. Këtu kemi kryer testimet apo matjet për rastet pa zgjidhjen për planifikim të transferimit të të dhënave.

Aplikime Android aktive apo në background janë përdorur gjatë matjeve sic do të flitet dhe me tej në këtë punim. Kryesisht kemi përdorur aplikimin Android PowerTutor, GSam, DroidWall, Google Play e Aperf etj si dhe për matjet fizike në UE me anë të matjes së rënies së tensionit në rezistencën seri ~0.22 ohm. Nga matjet Ekrani dhe ndërfaqja 3G/4G zënë një pjesë të konsiderueshme të konsumit të energjisë. Ndërkohë profili i energjisë sikurse shihet në figura ka raste kur UE shkon direkt nga PCH në DCH, nga PCH në FACH e më pas në DCH, si dhe për shkak të DRX/DTX mund të shihen dhe tentativa të kalimit në nivele më të ulët nga DCH por që shumë shpejt shkon në DCH.

Figura 5.24_2: Informacion mbi matjet me Samsung S3/4 GT-I9100 nga NetInfo II Api

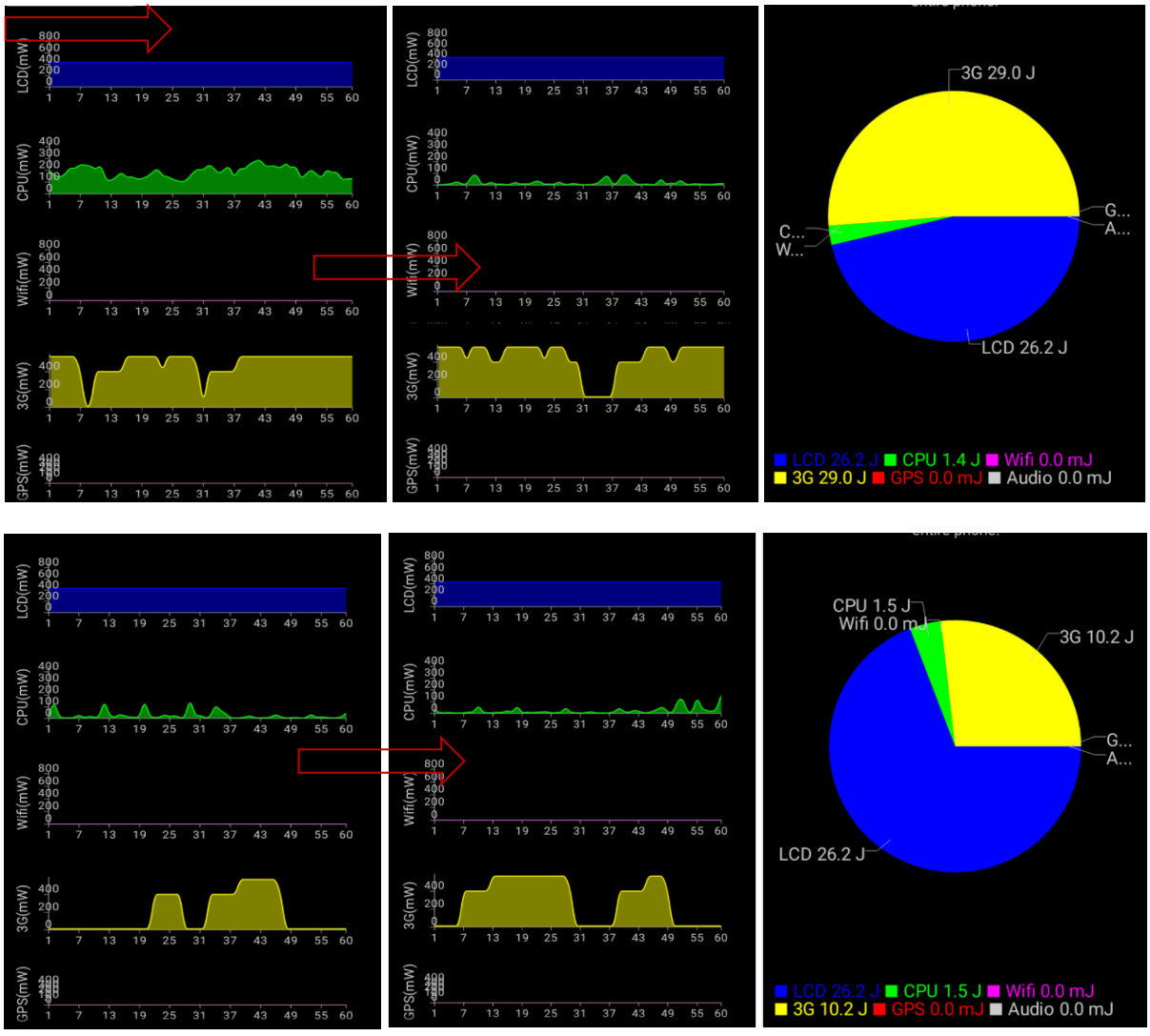


Figura 5.24_3: Rezultate matjesh në Samsung S3/4 GT-I9100 me PowerTutor (pa zgjidhjen)

Vihet re së me mbarimin e kohëve të pasivitetit kemi një kalim nga DCH (> 600 mW) në FACH (~ 400 mW) e më pas në PCH ku konsumi është pothuesje në ~ 0mW. Për kohët e pasivitetit me sa mund të thuhet se T1 = 2 sek dhe T2 = 5 sek:

Kanali	Energjia
PCH	~0 mW
FACH	~ 400 mW
DCH	> 600 mW

Tranzitimi	Koha pasivitetit
DCH => FACH (T1)	~ 2 sek
FACH = > PCH (T2)	~ 5 sek

Tabela 5.7 : Te dhena nga matjet me smartphone me OS Android

Sikurse mund të shihet ndërfaqja e rrjetit në UE gjatë komunikimit krijon një konsum të shpejt të energjisë prandaj menaxhimi i konsumit duhet të menaxhohet në rregull.

5.8 Modeli matematikor i konsumit të energjisë së UE/smart

Për modelimin e konsumimit të energjisë në një telefon UE ne duhet të marrim disa informacione si të dhëna hyrëse. Kjo e dhënë esenciale është trafiku i hyrjes me kohën e saktë (timestamp). Empirikisht, është demonstruar se fuqia e konsumuar në një UE mund të ndahet në shkallë ose nivele. Kur një UE merr një paketë UE vendoset në gjendje në nivel me fuqi të lartë, dhe kur nuk ka paketa të marra gjatë ndonjë timeout-i, UE ndryshon gjendje në një nivel tjetër më inferior dhe kështu deri në një nivel fuqie më të ulët. Për modelimin e fuqisë së konsumuar në një UE ne duhet të krahasojmë nivelet e fuqisë më një shumë fuqie të konsumuar dhe një kohë "timeout" midis tyre. Por ka një problem, pasi që ka mundësi që UE të ndryshëm të tregojnë sjellje të ndryshme të fuqisë, por në parim sjellja apo grafiku i konsumit të energjisë është i ngjashëm.
Ka 4 nivele fuqie, ku vlerat empirike të cdo niveli jepen në tabelën 5.3 e 5.4. Këto 4 nivele fuqie mund të shkaktohen si shkak i shpejtësive të ndryshme të aksesit në çdo nivel. Bazuar në tabelat e përmendura si dhe shpjegimet respektive të fuqive të kërkuara, madhësisë së të dhënave, shpejtësive të ofruara për kanalet dhe kohëzuesve të pasivitetit rezultojmë në një model të përgjithshëm matematikor grafikisht si në figurë:

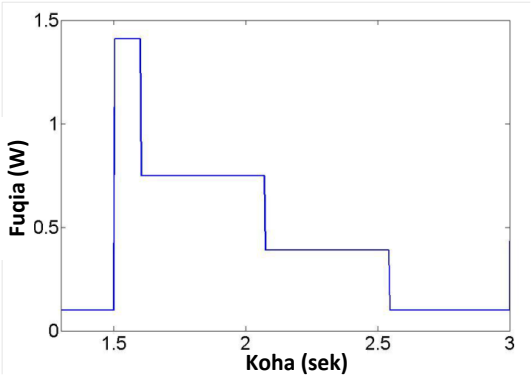


Figura 5.25: Modelimi i niveleve të fuqisë për një komunikim të dhënash në 3G (shembull)

Pra, për çdo komunikim një ndërfaqe rrjeti 3G do të kalojë nëpër një nivel energjitik të tillë sikur në figurën 5.25 (e më në detaje figura 5.26), kur të dhënat kryesisht janë në një volum të madh që lejojnë tranzitimin në DCH direkt (pra Idle → DCH e më pas në DCH → FACH → Idle).

Tranzitimi nga një nivel fuqie në një tjetër varet nga faktorë të ndryshëm. Varet se në ç'nivel ai është, sa sekonda kanë kaluar që nga marrja e paketës së fundit dhe nëse në një moment është duke marrë një paketë apo jo etj.
Në mënyrën FD (Fast Dormancy) ku shpesh operatorët mund të veprojnë, nuk ka energji Tail (Bisht) të kërkuar përderisa ky shërbim (feature) vepron kështu dhe kujdeset për "prerjen" e bishtave dhe e mban telefonin UE në një gjendje fuqie më të madhe se sa ajo e Qetë (Idle).

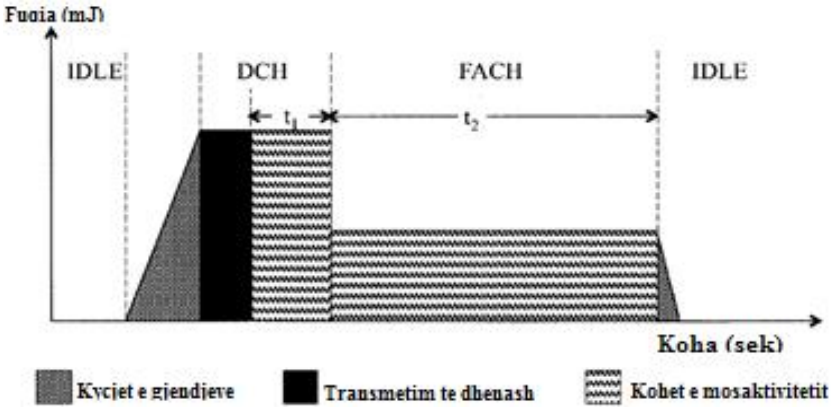


Figura 5.26: Modeli i thjeshtuar i konsumit të energjisë në 3G/UMTS

Megjithatë, nëse transmetimi i paketave është i shpeshtë më pas kostoja e energjisë së ngritjes (ramp up) është më e lartë për tu kyçur nga Idle në DCH. Koha totale e bishtave shprehet si $T_T = T_1 + T_2$. Por nëse ne rrisim kohën e bishtit energjia totale e konsumit do të bëhet më e lartë në mënyrën T_T si rrjedhojë e shpenzimit të energjisë në Tail (bisht).

5.8.1 Përcaktimi i modelit të fuqisë bazuar në mënyrën e operimit

Shumë komponentë hardware janë në gjendje për të punuar në disa gjendje të energjisë që korrespondojnë me nivele të ndryshme të konsumit të energjisë. Gjatë kohës së veprimit, energjia e gjendjes e UE përcaktohet nga aktivitetet që janë kryer dhe kapaciteti i përpunimit të komponentit hardware në çdo moment të caktuar. Nga një këndvështrim software, çdo komponent hardware ka disa mënyra operimi, që korrespondojnë me aktivitete të ndryshme dhe kapacitete përpunuese. Me fjalë të tjera, secila prej këtyre mënyrave operuese mund të ketë një një fuqi gjendjeje në nivel fizik. Duke pasur parasysh një mënyrë operimi, është e mundur që të nxjerrin konsumin e energjisë të komponentit hardware.

Në këtë nënseksion, ne tregojmë **modelimin e energjisë së një komponenti hardware bazuar në analizën e mënyrës së saj operuese.** Ne së pari ne duhet të japim një formë të përgjithshme të modelit të energjisë. Siç shihet në ekuacionin 5.13, energjia e konsumuar e një komponenti hardware mbi një kohëzgjatje është e përbërë nga energjia e shpenzuar në çdo mënyrë operimi dhe nga "mbingarkesat" e shkaktuara nga tranzicioni midis mënyrave operuese:

E = Energjia në çdo mënyrë operimi + "mbingarkesat"

(të shkaktuara nga tranzicioni midis mënyrave operuese)

$$E(t) = \sum_{0}^{j} \mathrm{Ej(tj)} + \sum_{0}^{j} * \sum_{0}^{k} \mathrm{Ej,k} * \mathrm{Cj,k}\,(t) \quad (5.13)$$

Ku:

- E(t) është energjia totale e konsumuar nga komponenti hardware përgjatë kohëzgjatjes t,
- tj është kohëzgjatja e shpenzuar në mënyrën operuese j.
- $E_j(t_j)$ është kostoja e energjisë gjatë t_j.
- $E_{j,k}$ është përcaktuar si "mbingarkesa" e energjisë e shkaktuar nga tranzicioni, nga regjimi operues j në k.
- $C_{j,k}(t)$ tregon se sa shumë e këtij tranzicioni ka ndodhur gjatë kohës t.

Energjia e shpenzuar në pritje për kalimin në një gjendje më të ulët të energjisë është quajtur shpesh energjia bisht. Mund të gjurmohet duke përdorur dy metoda. Njëra është për të lexuar direkt nga OS i UE. Për shembull, një profilizues i gjerë i energjisë së rrjetit për pajisjet e rrjetave, përshtat metodën e aplikimit Quanto. Megjithatë, Quanto kërkon ndryshime në driverat e pajisjes në mënyrë që driverat të mund të vënë në dukje gjendjet e energjisë se komponentëve hardware, themelore gjatë kohës së punës. Mënyra tjetër është të nxjerrin mënyrën e operimit nga një përshkrim i ngarkesës së punës në bazë të rregullave të tranzicionit. Rregullat e tranzicionit përcaktojnë kur kalojmë nga një mënyrë operuese në një tjetër. Në bazë të tyre, t_j dhe $C_{j,k}(t)$ mund të parashikohen/përcaktohen. Në literaturë, fuqia e një komponenti hardware është supozuar zakonisht të jetë afërsisht konstante në çdo gjendje të energjisë. Bazuar në këtë supozim, $E_j(t_j)$ është produkt i fuqisë konstante dhe t_j. Kur supozimi dështon, $E_j(t_j)$ mund të jetë një funksion i konsumit të energjisë me mënyrën e operimit, përshkrimin e ngarkesës së punës dhe kohëzgjatjen si variabla.

$E_{j,k}$ varet nga karakteristikat fizike dhe supozohet zakonisht si konstante. Vlera e $E_{j,k}$ mund të matet duke përdorur metodat e prezantuara më parë. Ndonjëherë, "mbingarkesa" e tranzicionit nuk llogaritet në konsumin e energjisë, sepse mbingarkesa e tranzicionit është mjaft e vogël dhe mund që të injorohet, ose monitorimi i tranzicionit nuk është i dukshëm.

Nisur nga kjo mund te behen dhe optimizime te konsumit te energjise si psh:

- *Reduktimi i kalimit në gjendje/kanale me nivel të lartë energjie.*
- *Reduktimi kycjeve në gjendje të ndryshme.*
- *Vonesa e transmetimeve në grup.*

5.8.2 Modelimi statistikor i Fuqisë bazuar në Utilizimin e Hardware-it

Metodat statistikore janë përdorur për modelimin e konsumit të fuqisë së komponentëve hardware. Variablat e modelit janë përcaktuar duke përdorur matjet e performancës, të cilat mund të reflektojnë përdorimin e komponentëve hardware. Përveç kësaj, një model i regresionit linear shpesh është përdorur si një model bazë, me koefiçientë të ndryshueshëm duke ju përshtatur

grupeve të të dhënave të mbledhura, duke përfshirë vlerat e variablave dhe të matjeve përkatëse të energjisë.

Në fakt, vlerat e performancës dhe vlerat që rrjedhin nga ato janë përdorur gjerësisht në modelet e fuqisë së mikroprocesorëve [46, 50, 88]. Këto vlera janë një grup i regjistrave të veçanta të ndërtuar në mikroprocesorët.

Ata ruajnë vlerat për aktivitetet e hardware të lidhura, sikurse numërimin e cikleve dhe numërimin e L_1 cache. Vlerat e këtyre "kounterave" / të dhënave mund të merren duke përdorur sisteme/mjete profilizimi si psh Oprofile 4 për sistemin Linux. Rezultati i modeleve mund të përdoret për të përcaktuar llogaritjet e kostove të aplikimeve të rrjetit, të tilla si kostoja e lexim/shkrim nga/në kujtesën e të dhënave. Përcaktimi i vlerësimit të modelit bazë të fuqisë mund të përdoret për menaxhimin direkt të energjisë si dhe analizën e efikasitetit të energjisë të zgjedhjeve të strukturës në fazën e projektimit software. Ekuacioni 5.14 tregon një formë të përgjithshme të një modeli linear, regresionit me variablat parashikues.

$$f(y_i) = \beta_0 + \sum_{j=1\ldots p}^{p} \beta_j g_j(x_{i,j}) \quad (5.14)$$

ku $g_j(x_{i,j})$ është një funksion para-procesimi i vlerës origjinale të variablit parashikues $x_{i,j}$. Duke pasur parasysh n vëzhgimet e vlerave të variablave parashikues dhe vlerat e përgjigjeve përkatëse $y_i (i = 1..n)$, vlerat e interceptuara β_0 dhe çdo koefiçienti g_j $(j = 1..p)$ janë rregulluar automatikisht gjatë përshtatjes së modelit drejt një modeli të ri, në të cilën përgjigja më e mirë mund të jetë e parashikuar nga variablat parashikues të modelit. Pasi modeli është ndërtuar, duke pasur parasysh vlerat e variablave parashikues, vlerësimi i konsumit të energjisë do të shfaqet.

5.9 Përmbledhje e problematikave të energjisë në Smartphone

Shumë analiza ilustrojnë faktin që në 3G e 4G energjia e komunikimit të përdoruesit fundor nuk ndikohet vetëm nga shuma e të dhënave të transferuara apo të marra, por gjithashtu dhe nga faktorë të tjerë si sinjali i marrjes RSSI, specifikime (settings) në background të telefonit, parametra statike të konfiguruar në rrjetin qendror (rrjeti radio), përditësime periodike të aplikimeve që punojnë nën backround/sfond (apo aplikimet gjithmonë në punë).
Për përdoruesit ka periudha kohore pa aktivitete (në lidhje me kohën, punën, lëvizshmërinë etj.) nga të cilat implementimi i disa kufizimeve mund të ndihmojë në përfitimin e reduktimit të energjisë.
Nëse përdoruesi apo koha e mendimit të tij është më e madhe se koha e pasivitetit, atëherë ky aplikim sjell një konsum të lartë të energjisë në çdo të kërkuar në web psh. Por në anën tjetër nëse aplikimi në mënyrë agresive bën "prefetching" të dokumentit dhe përdoruesi nuk kërkon ndonjë dokument të bërë "prefetch", përsëri aplikimi shpenzon një kohë dhe energji të konsiderueshme në *prefetching*.
Në lidhje me shërbimet e zërit, është parë që duke qenë i lidhur në rrjet, 3G shkakton një konsumim më të lartë të energjisë sesa 2G dhe shpesh deri në 50% më shumë. Ne 4G kjo eshte nje dicak tjeter sepse aty flitet per Voip apo VoLTE dhe kosnumi eshte baraz me energjinë e kërkuar për datat e transferuara.

Në anën tjeter kur të dhënat kërkohen të transmetohen, 3G ofron shpejtësi më të larta dhe konsum më të vogël në terma të energjisë për bit (dhe kohë). Shumica e përdoruesve kanë mundësi të zgjedhin rrjetin, e cila i jep mundësinë të kalojnë nga një rrjet te tjetri në varësi të shërbimeve aktuale që i duhen përdoruesit.
Një zgjidhje tjetër do të ishte të zgjidhej ndërfaqja ajrore që është më efiçente në energjinë e transmetimit. Psh 2G dhe 3G janë më shumë konsumuese sesa Bluetooth dhe WiFi kur bëhet fjalë për transferim dhe marrje të dhënash. Ndërfaqja Wlan qëndron në krye në këtë rast.
DL përkundrejt UL**,** ku **s**tatistikisht nga studimet e bëra shihet se shkalla e trafikut në UL është më e vogël se shkalla e trafikut në DL. Mesatarja e përgjithshme për DL dhe UL është në shkallën 6:1. Kjo asimetri, tregon që ka një ndikim më të lartë të trafikut DL, e cila duhet të merret në konsideratë në përgatitjet për teknologjitë e aksesit për smartphonët.
Por transferimet smartphone janë kryesisht të shkurta. Transferime të tilla kanë një kosto të lartë të energjisë dhe të amplifikojnë overheadet e protokolleve të shtresave të ulëta. Agregimi në grup (batch) i transmetimeve është një zgjidhje. Përdorimi i një proxy në "cloud" mund të lehtësojë këto transferime në grup.
Kanali PCH ka një konsum pak më të lartë sesa gjendja e Qetësisë/Idle por kërkon më pak mesazhe sinjalizmi për të kaluar nga një gjendje PCH në një gjendje DCH (12 mesazhe me 30 krahasuar nga gjendja Idle) sikurse është treguar më parë.
Fortësia e sinjalit ka një ndikim direkt në konsumimin e energjisë radio, e cila është një komponent domethënës në energjinë totale të energjisë së konsumuar të pajisjes UE. Për shumë studiues këto ndryshime në sinjal mund të shfytëzohen duke përzgjedhur komunikimin kur sinjali është më i fortë, dhe kështu energjia mund të ruhet deri diku.
Për shumë autorë e studiues pas testimeve të tyre është parë që evitimi i dritës së ekranit mund të jetë një ruajtës i mirë i energjisë dhe për më shumë motivon në përdorimin e dritës së ambjentit me përcaktimin e sensorëve të dritës në pajisjet pa tel që të mund të asistojnë në zgjedhjen e duhur të ndriçimit të ekranit. Për shembull zgjedhja e një drite më të errët mund të reduktojë konsumin e energjisë. Reduktimi i dritës gjatë një thirrjeje, dukshëm është një zgjidhje që redukton konsumin e energjisë gjatë një thirrjeje telefonike. Zbulimi i ndryshimit të gjendjeve apo RRC në rrjetin 3G dhe planifikimi i transmetimit në FACH janë ndër idetë e sotme në zhvillim e sipër. Ndërkohë në i shtojmë kësaj dhe një timeout në transmetim, dhe pasi ai mbaron kemi transmetim në grup. Le të shohim ndikimin e secilës pikë që përmendëm mësipër;

5.9.1 Ndikimi i protokollit RRC

Tranzitimi nga Idle në gjendjen DCH dhe tranzitimi midis Idle dhe gjendjes FACH janë të trigeruara nga aktiviteti i përdoruesit, dhe *gjendja që UE do të tranzitojë varet nga niveli i okupimit të Bufferit (shuma e të dhënave) në nivelin e shtresës RLC (kontrollit të radio linkut).* Bufferi i RLC është përdorur për të trigeruar tranzitimin e gjendjeve që mund të jetë DL apo UL. Kur përmbajtja e të dhënave në buffer tejkalon një vlerë të caktuar, sinjalizmi korrespondues bëhet përpara ndryshimit të gjendjes. Këto buffera janë pastruar pasi të dhënat transmetohen. Surprizisht, fuqia e konsumuar në FACH nuk rritet kur RSSI është e dobët, e cila bën që gjatë një thirrjeje të evitohet tranzitimi i gjendjes në DCH përkundrejt atij FACH sa më shumë të mundet.

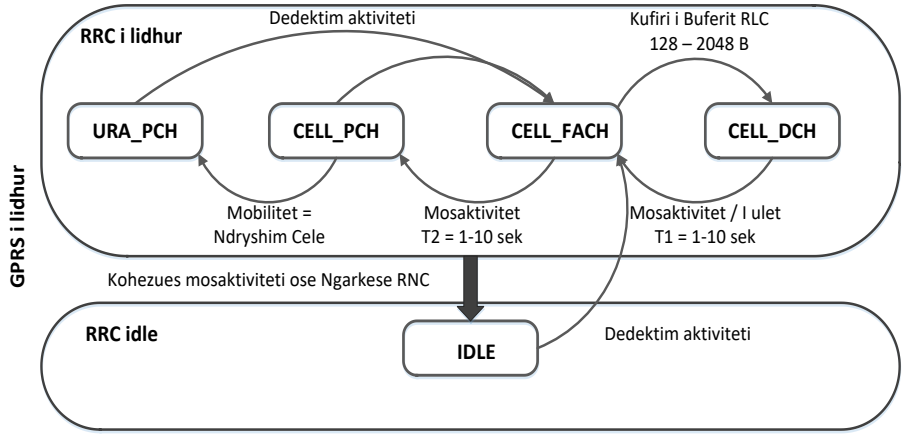


Figura 5.27: Gjendjet operacionale të UE dhe vlerat e kohëzuesve të pasivitetit

Sikurse shihet, impakti i parametrave të rrjetit sikur Tail (bishti), bufferi i të dhënave RLC ose kualiteti i radio linkeve në konsumimin e energjisë në UE është i lartë. Megjithatë, këto parametra nuk janë të mundshme për shtresat e aplikimit. Ekzistojnë algoritme për të nxjerrë timerat T_1 e T_2 (kohëzuesit e pasivitetit si në figurë) janë bazuar në njohjen që çdo gjendje ka në kompetencat e saj një RTT (kohë vajtje-ardhje) të ndryshme për një paketë të dhënash. Për të llogaritur RTT, është e nevojshme të dërgohen paketa testimi me një madhësi të paracaktuar nga UE. Një server i përgjigjet me echo-t duke lejuar llogaritjet e RTTs. FD'ja e cila e çon ndërfaqen radio të tranzitojë shpejt nga një kanal / gjendje DCH në një gjendje/kanal me energji më të ulët (Idle ose PCH).

5.9.1.1 Promovim dhe demontim i gjendjeve (UL / DL)

Qian at al. [13] shpjegon se dy "peer-e" entitetesh janë gjithmonë të sinkronizuara nëpërmjet kanaleve të kontrollit përveç gjatë ndonjë rasti dhe situatave të gabimeve. Gjithashtu shihet se kanalet DL dhe UL përdorin të njëjtin mekanizëm (të paktën në 3G). Ndryshimi (ulja) e gjendjes (në një kanal me energji më të ulët) konsiston në këtë rrugë: DCH→FACH, FACH→IDLE, dhe DCH→IDLE, shkon në drejtim të kundërt. Në varësi të gjendjes fillestare, një promovim gjendjeje është trigeruar ose nga një fillim aktiviteti transferim-marrje të dhënash nëse UE është në gjendje Idle/Qetësie, ose madhësia e bufferit të RLC tejkalon një kufi limit në çdo drejtim nëse UE është në kanalin FACH. Në realitet ulja/ndryshimi i gjendjes në kanale trigerohet nga 2 kohëzuesa jo-aktiviteti të mirëmbajtur nga nyja RNC në 3G dhe BSC në 2G (pasi transmetimet e të dhënave janë përfunduar).

Promovimi i gjendjes përfshin më shumë "punë" sesa Demontimi. Promovimi i gjendjes shkakton një vonesë "ramp-up" deri në 2 sekonda, gjatë të cilës disa dhjetra mesazhe kontrolli shkëmbehen midis UE dhe RNC e serverave të shërbimit për alokimin e burimeve sikur në figurën 5.28. Konkluzioni për promovimin e gjendjeve përcakton një nga 2 mënyrat e promovimit të adoptuara nga UMTSi P_1: IDLE→FACH→DCH, ose P_2: IDLE→DCH. Për uljen e gjendjeve/demotion janë përcaktuar 2 parametra D_1: DCH→IDLE, D_2: DCH→FACH→IDLE. Gjithashtu Min (Q_2 / P_V) dhe Max (Q_1 / P_M) paraqesin madhësinë e bufferave të RLC për mos-trigerimin dhe trigerimin e ndryshimt të gjendjeve. Dërgimi i një pakete UDP apo TCP mund të shkaktojë dhe trigerimin e një gjendjeje promovuese si psh nga IDLE→DCH.

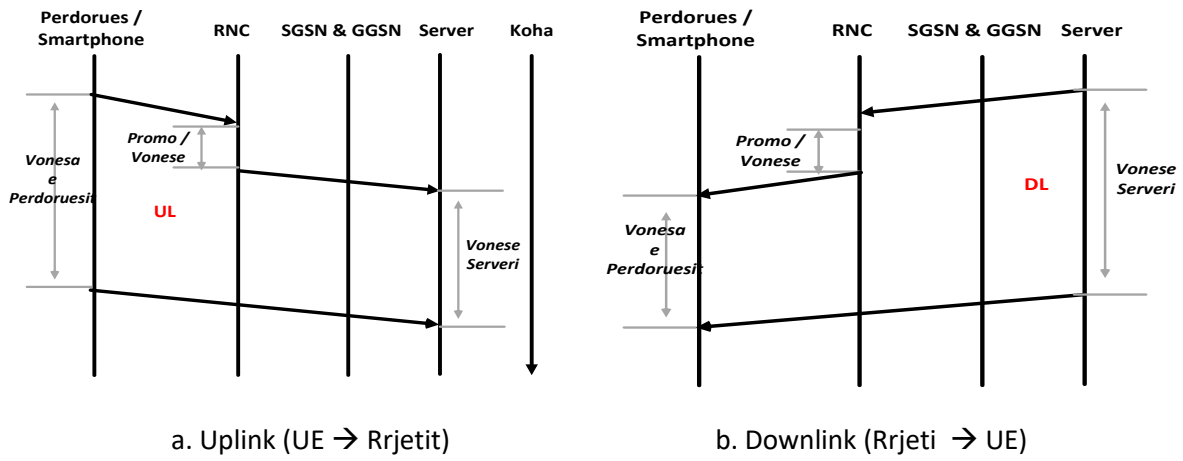


Figura 5.28: Promovim i gjendjes, i trigeruar nga një paketë në UL dhe DL

Energjia e transferimit në UL është më e madhe sesa DL për të dhëna me madhësi të madhe. Kjo diferencë është sepse bandwidthi i UL është më i vogël se sa bandwidthi në DL dhe kjo sjell zgjatje kohe për transferime të njëjta.

5.9.1.2 Bufferi i të dhënave në RLC dhe kohëzuesit

Sikurse për Vergara et al [55] bufferi i të dhënave të RLC's përcakton kufirin për tranzitimin e gjendjeve. Sipas analizës ekzistojnë 4 kufij, dy për tranzitim PCH-DCH dhe dy për FACH-DCH (në UL dhe DL). Po ashtu përdorimi i RTT është bërë për të nxjerrë në dukje gjendjen e UE duke dërguar një paketë të vogël testuese dhe dëgjuar për echo nga një server. Për nxjerrjen e kufirit të të dhënave për kalimin PCH-DCH, ne mund të përcaktojmë kufirin e bufferit duke vendosur 2 parametra një për nivelin e lartë/max e një për nivelin e ulët / Min (MaxBytes dhe MinBytes). Gjithashtu përdoren kohëzuesit T_1 e T_2. Për nxjerrjen e kufirit të bufferit për kalimin FACH-DCH, i afrohemi duke përdorur një paketë testuese më madhësi të madhe. Në fazën e dytë përdoret një paketë me madhësi më të vogël në mënyrë që t'i përafrohemi kufirit të paracaktuar dhe vazhdohet në këtë mënyrë derisa të kemi një paketë me madhësi sa kufiri i bufferit për ndryshimin e gjendjes apo bufferit/memorjes. E gjithë kjo është në përputhje me metodën 1 të mënyrës së nxerrjes së parametrave të rrjetit.

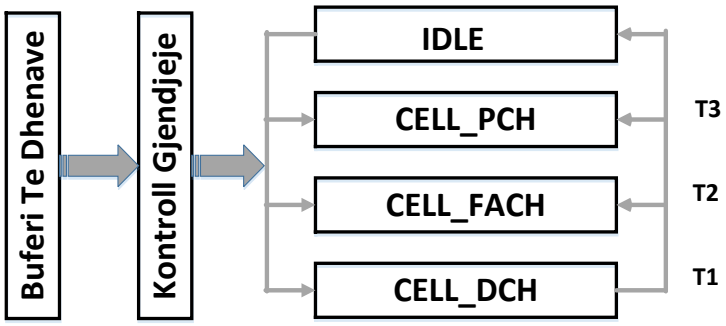


Figura 5.29: Ilustrim i bufferit dhe gjendjeve "makinë" RRC.

Tranzitimet e gjendjeve pikë së pari kontrollohen nga llogjika e ndryshimit të gjendjeve dhe janë të bazuara në shumën e të dhënave që do shkëmbehen. Kur bufferi është bosh, një gjendje aktuale

ndryshohet bazuar në kohëzuesit T_1, T_2 e T_3 (ku T_3 jo ndonjë domethënie për nga energjia dhe prandaj nuk konsiderohet). Energjia e konsumuar në fazën e transmetimit mund të llogaritet si:

$$E = T * U * I \quad (5.15)$$

ku ***T*** është vlera e kohëzuesit RRC, ***U*** është tensioni i kërkuar, dhe ***I*** rryma e harxhuar. Siç shihet E është kryesisht e përcaktuar nga vlera në sekonda e kohëzuesit RRC (në kohën kur bufferi është bosh).
Një gjendje nga PCH te DCH është trigeruar kur volumi i bufferit radio ***C*** tejkalon vleren B_1 dhe zgjat deri sa të gjitha kanalet e dedikuara janë lëshuar, ndjekur nga një kohëzues joaktiviteti T_1, dhe alternativisht që një kohëzues i aktivitetit të ulët ka vepruar për një kohë *Td*. Më pas ndërfaqja radio kalon në kanalin (gjendjen) FACH ku ajo qëndron derisa një kohëzues joaktiviteti tjetër T_2 vazhdon deri në mbarim dhe ajo shkon në kanalin PCH; veç në mos *C* tejkalon B_2 për të cilin një ndryshim gjendje në DCH trigerohet. Një trigerim nga PCH në FACH kryhet kur aty ka të dhëna të bufferuara/memorie por shuma është ende më e vogël sesa B_1. Duhet të themi se jo të gjithë kanalet dhe radiot implementojnë këto "feature" (shërbime të reja).

Më pas, ne përshkruajmë demontimin e gjendjeve, dmth, kalimin në gjendjet e mëposhtme me kapacitete më të ulëta apo performancë të ulët, dhe promovimin e gjendjeve që i referohen tranzicionit në një gjendje ku UE përjeton performancë të lartë.

Demontimi i gjendjes: Për çdo pako P_i në gjurmim dhe kohën (timestamp) të saj $t(P_i)$, llogarisim $\Delta_i = t(P_i) - t(P_{i-1})$, si kohën e kaluar në mes të paketës dhe paraardhësit e tij. Δ_i është përdorur për të simuluar progresin në kohëzuesit e pasivitetit/pasivitetit fikse T_1 dhe T_2: nëse $\Delta i > T_1$ ose $\Delta i > T_2$, ne do të trigerojmë ndryshimin përkatës të gjendjeve.
Me V në figurë i referohemi kapacitetit apo volumit të ardhur të paketave në ndonjë nga 2 drejtimet.

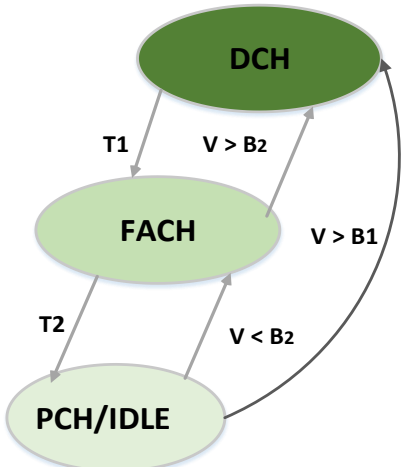


Figura 5.30: Kushtet e tranzitimit të gjendjeve makine në 3G

Kujtojmë se disa rrjeta zbatojnë një mekanizëm të ulët të aktivitetit në DCH për lirimin e kanalit të transportit dhe kalimit në FACH para se kohëzuesi i pasivitetit T_1 të skadojë [102]. Në mënyrë që të stimulohet ky mekanizëm i ulët i aktivitetit, ne përcaktojme T_d si periudha e kohës gjatë së cilës monitorohet shuma e të dhënave të dërguara. Ne përmbledhim madhësinë e paketave gjatë T_d dhe të detyrojmë UE të shkojë në FACH kur të dhënat e dërguara janë më të ulëta se një

prag/kufi limit D_d / B_2. Kjo është e njëjtë si duke përdorur një prag mbi vlerën aktuale bit-rate që është përdorur në mekanizmin standard.
Më në fund, në mënyrë që të japin llogari për sinjalizimin në mes të gjendjeve, çdo demontim gjendjeje ka parapërcaktuar një kohëzgjatje tranzicioni e cila i është shtuar kohëzuesve të pasivitetit (p.sh., $T_1 + T_{DCH\text{-}FACH}$ dhe $T_2 + T_{FACH\text{-}PCH}$ / Idle).
Promovim gjendjeje: Promovimet e gjendjeve janë të kontrolluara nga katër pragjet/kufijtë e bufferave/memorjeve RLC: Dy uplink (M^u_1 dhe M^u_2) po mund të shënohen si M_1 dhe M_2 (me tej dhe si P_M e P_V) dhe dy downlink (B^d_1 dhe B^d_2). Kur UE është Idle/ PCH, ne përdorim drejtimin e paketës Pi (uplink ose downlink) për të krahasuar madhësinë e saj me pragjet korrensponduese përkatëse M^u_1 dhe M^d_1. Nëse madhësia e P_i është më e madhe se pragu përkatës, UE është lëvizur direkt në kanalin DCH. Përndryshe, UE është zhvendosur në kanalin FACH.
Simulimi i promovimit të gjendjeve në FACH është kryer nga okupimi i bufferit RLC. Ne kemi simuluar okupimin aktual V për çdo drejtim (V^u dhe V^d), duke marrë parasysh paketat që duhet të dërgohen dhe kapaciteti i të dhënave të kanalit. Kjo është bërë si vijon: derisa madhësia dhe drejtimi (uplink ose downlink) për çdo paketë Pi është i njohur, *V* është llogaritur duke shtuar madhësinë e P*i* në byte për bufferin korrespondues. Pastaj, kur $V > M_2$ apo B_2 në çdo drejtim, UE ka lëvizur nga FACH te DCH.
Të dhënat transmetohen (dmth, bufferi është zbrazur dhe kemi vendosjen e $V = 0$) në varësi të shpejtësisë së të dhënave kanal. Së pari, përcaktohet Δ^u_i dhe Δ^d_i si kohën e kaluar nga paketa e fundit uplink apo downlink (në të njëjtin drejtim), që llogaritet nga gjurmimi i paketës. Duke pasur parasysh *V* dhe shkalla e shpejtësisë së të dhënave të kanalit, *koha për të zbrazur bufferin RLC (T_e) mund të llogaritet* (dmth, kur të dhënat dërgohen në të vërtetë). Ngjashëm me pragjet RLC, T_e varet edhe nga drejtimi: T^d_e dhe T^u_e për downlink dhe uplink respektivisht.
Bufferi është zbrazur kur koha për të zbrazur bufferin është më e madhe se koha e ndërpaketës në të njëjtin drejtim: $V^u = 0$ nëse $T^u_e > \Delta^u_i$ për bufferin uplink dhe $V^d = 0$ nëse $T^d_e > \Delta^\Delta i$ për bufferin downlink. Modeli lejon të specifikojmë T_e duke supozuar shpejtësinë konstante të të dhënave (p.sh., $T_e = K / 512$ kbps) ose si një funksion i *V* në bazë të matjeve të shpejtësisë së të dhënave në FACH për një rrjet të vërtetë. Më tej në punim parametrit T_e do ti referohemi si K_p.
Parametrat për modelin 3G e 4G: Të dy T_1 dhe T_2 (kohëzuesit e pasivitetit) të një game rrjetesh të vërtetë janë në dispozicion të literaturës në [24, 26, 152, 177, 198] ose mund të merren ose nxirren nga një matje e vetme e energjisë kur jemi duke dërguar një paketë të vetme dhe duke vëzhguar kohën e shpenzuar në çdo gjendje. Në mënyrë të ngjashme, kohët e kohëzgjatjes së tranzicionit janë marrë nga një matje e energjisë gjatë dërgimit të të dhënave të gjeneruara nga aplikimet e ndryshme (p.sh. vonesa të vërejtura në, ose të matur në kohën vajtje-ardhje/RTT për tranzicionet e ndryshme duke dërguar ping-e të ndryshëm në gjendje të ndryshme. Si një thjeshtësim, një vlerë e vetme është miratuar/përshtatur për çdo kohëzgjatje e tranzicionit/gjendjes. Çdo vlerë është vendosur nga një mesatare mbi 10 teste matjesh të energjisë.

5.9.1.3 Teknika të planifikimit të të dhënave

Për të gjitha sa u tha më siper lind nevoja e optimizimit të transmetimeve data në komunikimet mobile. Sikurse është përmendur dhe ne kapitullin për punime në këtë fushë, propozimet e testimet janë të shumta por gjithmonë është lënë vend për përmirësime të mëtejshme.

Kështu autorët te [55] prezantojnë një algoritëm të transmetimeve të të dhenave, që përdor parametrat (madhësinë e bufferave RLC për ndryshimin e gjendjes) të cilat duhet të jenë nxjerrë më parë. Qëllimi i algoritmit është të minimizojë energjinë e konsumuar duke zgjedhur paketa të vogla të dhënash në gjendjen FACH dhe shmangi koston e tranzitimit në DCH që drejton /çon në ekstra energji të DCH Tail dhe FACH Tail. Kur UE është në gjendjen DCH do të transmetojmë paketat e ruajtura në disa memorje, për paketat e mëdha dhe për paketat e vogla (ose me sipër i jemi referuar dhe me Mx). Çdo paketë e re që vjen, futet në pritje në varësi të madhësisë dhe klasifikimit të bazuar në afatin kohor më të afërt. Kur UE është në FACH, ajo transmeton gjithë paketat e ruajtura në bufferin e vogël, duke anashkaluar mundësinë e ndryshimit të gjendjes në DCH. Kjo arrihet duke dërguar një numër të paketave ku shuma e të cilave është pak më e vogël në bytes se kufiri i bufferit. Pastaj UE pret për kohën e pritjes apo të vonesës së dërgimit përpara dërgimit të paketave të radhës. Kjo mundëson që UE të mbetet në gjendjen FACH deri kur bufferi i vogël është boshatisur. Megjithatë dhe kjo metodë e implementuar krijon një sërë problemesh për komunikimet TCP pasi është me së shumti e bazuar mbi atë UDP porse në komunikimet reale shumë prej aplikiemeve përdorin protokolle si TCP që nuk janë shumë neglizhente ndaj vonesave. Për këtë gjë në implementojmë këtë zgjidhje por duke marrë në konsideratë dhe një sërë parametrash sikurse i përmendim dhe te seksioni 6.3.4.2 me poshtë. Poashtu dhe duke konsideruar faktorin e përgjumjes apo kalimit në PCH kur nuk ka aktivitet në UE.

Duke i analizuar më pas çdonjërin prej tyre ne mund të shikojmë se si përdoruesi apo rrjeti ka impakt në energjinë e konsumuar në një pajisje pa tel. Së pari është energjia e transmetuar që është proporcionale me gjatësinë e transmetimit (të dhënat) dhe nivelin e fuqisë së transmetuar. Së dyti në 3G protokolli RRC – kontrolluesi i burimeve radio që është përgjegjës për alokimin e kanaleve dhe shkallëzimin e energjisë së konsumuar nga radio bazuar në kohëzuesit e pasivitetit sikurse për autorët te [2, 4]. Bishti (tail) përkundrejt tranzitimit nga niveli i gjendjes së lartë në të ulët të fuqisë pasi një paketë është transmetuar, tranzitimet e pajisjes UE smart ndodhin vetëm kur rrjeti është joaktiv për një periudhë dhe kohëzuesi i joaktivitetit ka mbaruar sikur në figurat e mësipërme.

Ky mekanizëm shërben për dy të mira: 1) *lehtëson vonesat* e shkaktuara gjatë lëvizjes nga një gjendje me fuqi të lartë në një gjendje qetësie dhe: 2) *redukton sinjalizimin* e shkaktuar prej alokimit të kanaleve dhe lëshimit gjatë tranzitimit të gjendjeve. Për të adresuar konsumin e lartë të energjisë si rrjedhojë e "bishtave" radio të gjatë (në kohë), shumë vendorë telefonash suportojnë disa shërbime të rinj të rrjetit të quajtur "Përgjumje e shpejt" FD, që mundëson një tranzitim të shpejt në gjendje të Qetë. Përdoruesi/pajisja UE smart thërret procesin FD duke përdorur kohëzues më të shkurtër mosaktiviteti për të shkurtuar bishtat radio. FD është arritur duke dërguar mesazhe specifike sinjalizimi të quajtur *mesazhe indikative* të lëshimit të lidhjes sinjalizuese (SCRI).

5.9.1.4 Transmetimi në grup i të dhënave për të minimizuar energjinë

Nga sa është parë duhet thënë se është më mirë të performohen disa transferime të njëhershme në një kohë te vetme me koston e një Tail-i dhe një Ramp-i sesa të transmetohen në kohë të ndara me shumë "Tail" e "Ramp" (Qian et al, 2010). Për $\boldsymbol{n}$ kërkesa të njëjta, ku çdo kërkesë $\boldsymbol{r_i}$ është e lidhur me një kohë-mbërritjeje $\boldsymbol{a_i}$ dhe një deadline $\boldsymbol{d_i}$ gjatë të cilës duhet të jetë transmetuar. Kur kërkesa $\boldsymbol{r_i}$ është planifikuar të transmetohet në kohën $\boldsymbol{s_i}$, ndodh tranzitimi radio në gjendje / nivel (energjie) të lartë, dhe mund të bëhet transferimi i menjëhershëm i kërkesave $\boldsymbol{r_i}$, dhe ndërfaqja

radio mbetet në gjendje të lartë për T njësi kohe, e njëjtë me kohën Tail (bishtit) pasi transferimi ka perfunduar.
Kur shumë kërkesa janë transmetuar në të njëjtën kohë, pajisja është në gjendje energjie të lartë vetëm për kohën T (njësi kohe). Një app Tailender i autoreve te [3] përdor një algoritëm online për të zgjedhur transmetimin e një kërkese $\boldsymbol{r_i}$ në kohën $\boldsymbol{t}$. Ideja kryesore e autorëve të TailEnder është të transmetojë një kërkesë $\boldsymbol{r_i}$ nëse: *a)* kërkesa mbërrin brenda një kohe **x*T** nga kufiri limit / deadline i mëparshëm $\boldsymbol{d_0}$, ose: *b)* kur deadline i kërkesave ka kaluar afatin limit.
Kjo llogjikë përdor nocionin e kohës më të largët për transmetimin e paketës për tu kompletuar si transmetim, por dhe si pika e fundit në të cilën një transmetim është inicuar. Në llogjikën e tailender, të gjithë paketat e të dhënave janë grumbulluar derisa të arrihet deadline-i (koha limit). Sapo *afati kohor* mbaron, të gjitha këto paketa transmetohen si një burst i vetëm dhe ne fund me nje T2. Kështu që UE do të kyçet në gjendje të lartë energjie vetëm një herë, në vend që të kyçet çdo herë për transmetim të dhënash.

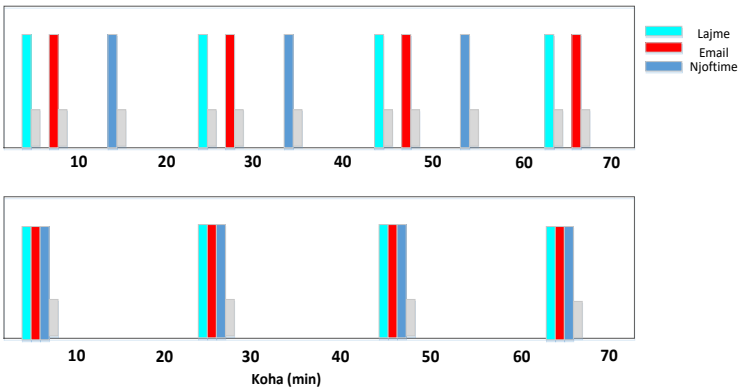


Figura 5.31: Ilustrimi i logjikës së planifikimit të transmetimeve

Në mënyrë intuitive është më mirë të kryhen disa transferime burst në një kohë me një kosto të vetme të *ramp* dhe *tail*, sesa të kemi disa cikle transferimi dhe ramp-e/ngritje apo tail-e/bisht. Për të treguar të mirat e kësaj metode ne përdorim këto vairabla dhe ekuacione që tregojnë qartësisht përfitimin ku:

- E_R është kostoja e energjisë për një ramp (zgjim).
- E_T është kostoja e energjisë për një tail (bisht).
- E_{50k} është kostoja e energjisë për një transferim 50 kb për paketë.

Kostoja totale e energjisë së n transferimeve të pavaruara në 3G do të jetë si:

$$n * (E_R + E_{50k} + E_T) \qquad (5.16)$$

Në rastin e aplikimeve të grupuara në transferim së bashku si grup (batch), kostoja e ***a*** *aplikimeve* në ***b*** *grupime (batch)* jepet si:

$$b * (E_R + aE_{50k} + E_T) \qquad (5.17)$$

Kështu psh. nëse do të kishim 9 transferime të pavaruara ne do të kemi një kosto energjie prej:

$$= 9E_R + 9E_{50k} + 9E_T$$

Dhe nëse do të kishim transferim në tre grupe (batch) atëherë energjia e kërkuar në transmetim do të jetë:

$$= 3E_R + 9E_{50k} + 3E_T$$

Pra ideja e transferimit në grup, sjell një reduktim të energjisë për të njëjtin numër aplikimesh. Në këtë mënyrë reduktohet dhe sinjalizimi i shkëmbyer si dhe kemi vetëm një bisht/tail për një grup transmetimesh të agreguara.

5.9.2 Niveli i sinjalit RSSI

Komunikimi në zona me nivel të dobët të RSSI mund të rezultojë në një shpejtim të "shterimit" të baterisë që shpesh mund të jetë dhe 50% më i lartë se sa në rastet me sinjal të fortë. Fortësi sinjali të moderuara (-90 deri -70 RSSI) janë më shpesh të hasura sesa ato ekstreme (min e max). Kështu si konkluzion kur një sinjal është i dobët jo vetëm që shpejtësia e të dhënave bie por dhe koha e transferimit do të jetë më e gjatë, sepse ndërfaqja radio është gjithashtu duke konsumuar më shumë energji. Kostoja e komunikimit në -90 dB në krahasim me -70 dB RSSI detyron përdorimin e një fuqie shtesë prej 20% (shiko figurën) dhe shpejtësia e të dhënave do të bie me 50% (ç'ka do të thotë dyfish i kohës). Sinkronizimi nëpër të dhënat në sfond mund të kryhet kurdo që sinjali në telefon të jetë i fortë p.sh 5 vija ose ~ -65dB RSSI.

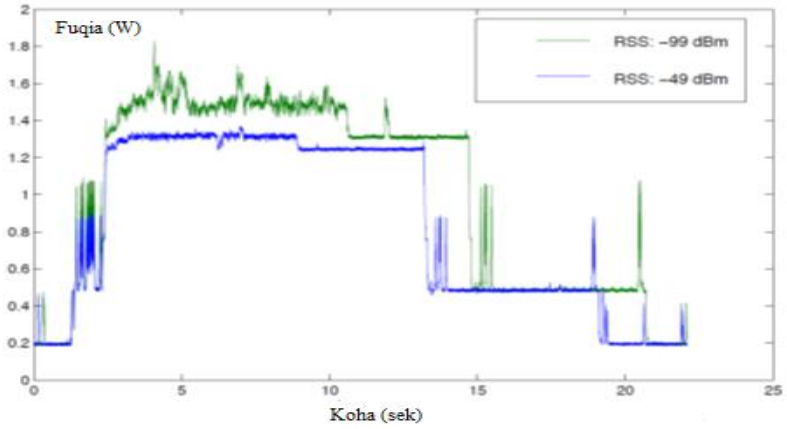


Figura 5.32: Konsumi energjisë për RSSI të ndryshëm [46]

Arsyeja pas kësaj rritje të energjisë është se për të mbajtur komunikimin "on" (aktive), amplifikuesi i terminalit UE rrit fuqinë e tij për të kompensuar humbjet për rënien e RSSI. Pra një kosto shtesë për nivele të dobët të RSSI. Gjithashtu do përdoren dhe skema moduluese me kontroll gabimesh më të larta (duke ulur nga ana tjetër dhe shpejtësinë reale të të dhënave). Pradaj dhe komunikimi për kanale në 3G si FACH e DCH është i ndryshëm për distanca të ndryshme të UE nga BTS/stacioni.

5.9.3 Modulimet dhe ndikimi i tyre te fuqia

Për të arritur transmetimin e të dhënave dixhitale nga ajri, bitet e të dhënave duhet të procesohen në simbole dhe pasi të mbarten në një valë mbartëse sinusoidale dhe kjo duke përdorur modulimin. Një shpjegim bazë i modulimit është të vendosë një marrëdhënie midis shumës së biteve në

simbole për të rritur shpejtësinë e transmetimit dhe përfituar një eficencë spektrale. Ka formate modulimi të ndryshëm, të përdorur në varësi nga shërbimi dhe gjendja e kanalit. Formatet më të përdorshëm të përdorur për telefoninë *mobile* janë QAM (modulimi në amplitute kuadraturë) dhe PSK (modulimi me zhvendosje në fazë). QPSK ishte modulimi i parë i përdorur në fillimet e para të 3G te WCDMA dhe mbart 2 bite për simbol dhe që mundëson një shpejtësi teorike deri në 2Mbit/s. Në 3GPP Release-t e mëvonshme të WCDMA, të quajtura HSDPA (aksesi i shpejtësisë së lartë të paketave në DL) është nxjerrë me modulimin 16QAM. Ky modulim mbart 4 bartëse për simbol dhe u mundëson një rritje të shpejtësisë së transmetimit në 7.2 Mbit/s. Ndër RNC SW release-t e fundit përdoret modulimi 64 QAM, e cila mbart 8 bite për simbol, suporton një shpejtësi teorike prej 84Mbit/s. HOM, apo modulimet më të larta kanë kontribuar në rritjen e shpejtësisë së të dhënave në teknologjinë 3G. Zhurmat në kanalet wireless shkaktohen nga transmetimi i kanaleve fqinje dhe ky është një nga faktorët më të rëndësishëm që duhet të merret në konsideratë në të ardhmen. Sikurse shihet dhe në figurë, pikat e konstelacionit janë shumë më të ngushta në modulimet më të larta.

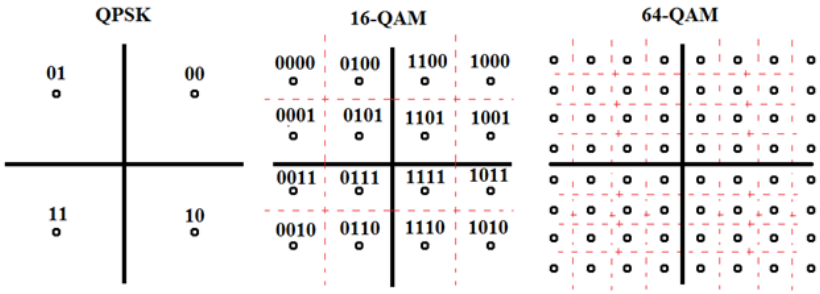


Figura 5.33: Konstelacioni dhe zonat e vendosjes të QPSK, 16- dhe 64-QAM

Në një ambjent me nivel zhurmash të lartë, simbolet kuptohen me vështirësi në marrës për tu dekoduar në mënyrë korrekte dhe kjo gjeneron një probabilitet të madh të të pasurit BER të lartë. Një koncept i termit të matjes në komunikimet dixhitale është raporti S/N për bit apo SNR ose e thënë ndryshe raporti $Eb/N0$ ku Eb është energjia për bit dhe N_0 është totali i densitetit të zhurmës. Sikurse mund të shihet dhe nga grafiku për një vlerë fikse të BER korresponduesi i raportit sinjal-zhurmë (SNR) rritet për modulimet më të larta (HOM) e cila gjithashtu provon që HOM janë në mënyrë sinjifikative më të ndjeshëm nga zhurmat.
Si konkluzion një modulim më i lartë kërkon më shumë energji për bit për të arritur të njëjtin BER sikur dhe për modulimet më të ulëta, kështu që duhen marrë në konsideratë në analizë përveç modulimit dhe kodimeve në kanal për të studiuar dhe arritur një eficencë energjie.

5.9.4 Ndikimi i ndërfaqes WiFi

Në krahasim WiFi shfaq një kosto të lartë fillestare të lidhjes me AP. Megjithatë WiFI në telefona përdor mënyrën PSM të energjisë, dhe kostoja e mirëmbajtjes së tyre është relativisht e vogël. Nga matje të ndryshme të shumë autorëve në këtë fushë theksohet se energjia e transmetimit nga WiFi është dukshëm më e vogël sesa 2G e 3G, specifikisht për transferime të dhënash të mëdha. Studiet e ndryshme kanë treguar që konsumimi i fuqisë është proporcional me ngarkesën e shkaktuar nga transferimi sesa nga transferimi total në vetvete. *Nëse më shumë të dhëna duhet të transmetohen, WiFi duhet të përdoret kur të krijohet mundësia.* Mbajtja e ndërfaqes WiFi aktive, kërkon një

energji për mirëmbajtje, sikurse është parë dhe nga shumë studime si dhe nga matjet tona. Megjithatë kyçja e WiFi në OFF pasi ka transmetuar dhe nëse nuk ka më të dhëna të tjera marrje-dërgim mund të ndihmojë në reduktimin e konsumit të energjisë por do të kemi përsëri një rritje të konsumit pasi kërkohet ndryshimi i gjendjes së ndërfaqes në OFF/ON dhe kjo në vetvete kërkon një fuqi.

Pra ndërfaqja 3G/4G sikurse shihet dominon në komunikime, dhe kalimi inteligjent (wifi on, skanim, wifi off)) kur mundet gjatë transferimeve në ndërfaqen WiFi ndikon në uljen e konsumit të shpejt të energjisë.

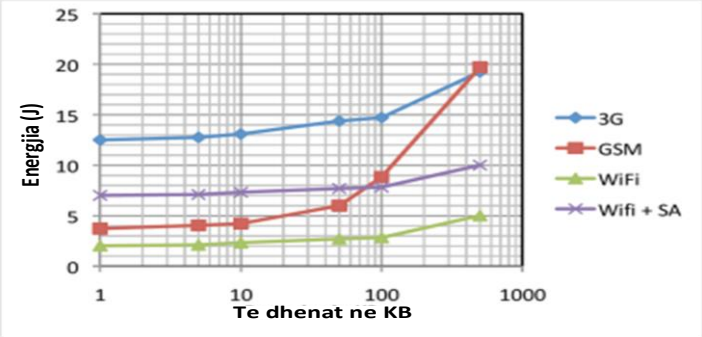


Figura 5.34: Energjia e konsumuar gjatë transferimit për ndërfaqet e ndryshme të UE [1,5]

5.9.5 Lidhjet paralele për të dhënat paketë (miks)

Zbulimi i aktiviteteve pikë së pari si dhe ri-incimi i atyre në pritje do të ishte fitimprurës sikurse përmend dhe Nurminen et al [27]. Kur shkarkimi të dhënave është kryer gjatë një bisede, konsumimi i fuqisë është rritur vetëm pak krahasuar me rastet kur biseda dhe transferimi i të dhënave do të kryheshin veç e veç. Nëse një transferim të dhënash kryhet gjatë një bisede telefonike, përdoruesi është në gjendje të kryejë të njëjtin aktivetet me vetëm 23% të energjisë që do t'i duhej nëse do të kryhej jo gjatë një bisedë telefonike (në 3G), pasi në 2G shërbime ekstra duhen në rrjet për të lejuar këtë gjë (DTM - dual transfer mode). Kyçja me forcë ose HO's konsumojnë energji ekstra, e cila duhet të merret në konsideratë në fuqinë totale të konsumuar në një pajisje *mobile.*

5.9.6 Koha dhe gjendjet e ekranit UE

Përdorimi i telefonave celularë dominohet zakonisht nga periudha të gjata të mosaktivitetit nga ana e përdoruesit, gjatë të cilave vetëm aktivitetet në sfond / background mund të zërë pjesë. P.sh ka periudha kohore si gjatë natës ose psh duke udhëtuar me makinë apo të zënë me punë të tjera ku pajisja UE nuk përdoret në *biseda* dhe *të dhëna* (përveç aplikimeve aktive në sfond/në background. Në këto raste mund të jetë fitimprurëse duke përdorur ekranian e telefonit dhe orën për të vendosur nëse do të hyjmë në një mënyrë të ruajtjes së fuqisë (apo PSM) ku kryesisht të dhënat nuk lejohen edhe në background, ose dhe kyçja midis 2G e 3G. Kështu që mund të kalohet nga 2G në 3G kur ekrani i UE bëhet aktive (nga prekja/touch) ose nga përdorimi i sensorëve të dritës dhe lëvizjes. Dhe pas njëfarë kohe joaktiviteti të mund të lejohet të kalojë në 2G (pasi 2G rezulton me konsum më të ulët të fuqisë) duke lejuar të dhënat ON ose jo për aplikimet në background etj, pa kërkesë për ndonjë shpejtësi të lartë të dhënash. Duke vendosur një kohëzues

pergjumjeje si T_{sleep} për periudhën kohore nga 23:00 deri në 06:30, e marrë si një shembull, dhe duke përdorur po një kohëzues/parametër T që kërkon në mënyrë periodike orën gjatë një 24 orarëshi. Në këndvështrimin e një përdoruesi nga ana e fuqisë së telefonit, do të ishte më me sens që ai të jetë gjithë kohës i lidhur në rrjetin 2G dhe të kalojë në 3G apo 4G vetëm nëse ka të dhëna me kapacitet të madh për të marrë e dërguar.
Duke krahasuar energjinë totale të konsumimit për transferim të dhënash gjatë ditës (mesatarizim) përkundrejt natës, siç është parë edhe pse Energjia e Ramp dhe Tail janë të njëjta si gjatë natës dhe gjatë ditës, energjia totale e konsumuar gjatë natës është më e vogël (me disa %) krahasuar për të njëjtin aktivitet gjatë ditës. Kjo besohet se vjen nga fakti i një konxhestioni më të ulët gjatë natës duke çuar në një transferim energjie më të ulët (kapacitete më të mëdha gjatë natës).

5.9.7 Orët e mbingarkesës (BH)

Sikurse është parë prej shumë testimeve edhe pse sinjali mund të jetë i fortë përsëri shpejtësia e të dhënave është e ulët dhe mundësitë më të mëdha janë: Rrjeti i zënë / BH ku trafiku i konsumuar është i lartë (piku i trafikut) ose në rrjet mund të jetë problem ndonjë nyje që i ka tejkaluar limitet ose ka probleme dhe krijon atë që quhet "bottleneck"/ ngushticë. Duke mos lejuar transmetimin e të dhënave kur shpejtësia është e vogël (në rastet e orëve të pikut BH) mund të rezultojë që të ruajmë një pjesë së fuqisë së pajisjes. Ka mundësi të ndryshme realizimi si p.sh dërgimi i një pakete *ping* dhe analizimi i paketës në marrje si dhe i TTL në ms. *Pingu* mund të iniciohet në vetvete si proces në background/sfond gjatë kohëve të paracaktuara (orët e pikut) duke mbledhur dhe analizuar përgjigjet dhe duke i përdorur në algoritmin bazë për ndërtimin e një zgjidhjeje "power save". Aktualisht shtyrja e transferimeve në këto raste do të ndryshonte/zhvendoste profilin e BH në operatorin mobile duke i zhvendosur ato në një periudhë tjetër kohore. Kjo duhet marrë në konsideratë në përdorimin masiv të kësaj logjike zgjidhjeje.

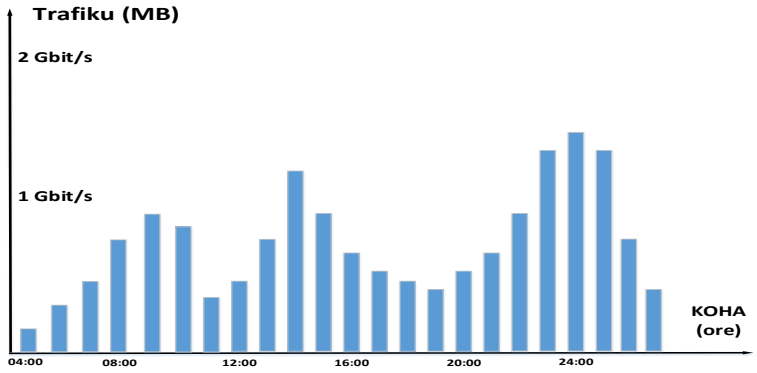


Figura 5.35: Profili BH i ndërfaqes IU-PS (RNC - SGSN/GGSN)

Megjithatë ekzistojnë dhe një sërë zgjidhjesh të tjera SW si kompresimimi i të dhënave para transmetimit duke ulur fluksin e të dhënave të transmetuara (në proporcion me kohën e transmetimit), proceset e Offload-imit të të dhënave në një Server apo PC, etj.

Në kohën e sotme ka dhe një sërë punimesh drejt zgjidhjeve HW (deri në nivel tranzistorësh) në smartphone në lidhje me modele të reja baterish, funksionalitete të ekraneve, ndryshimit të shpejtësisë së procesimit të CPU've etj.

5.10 Vlerësimi energjitik ne kanale

Vlerësimi energjitik – rasti DCH & FACH në vecanti

Konsiderata 2: Trafik i ulet të dhënash – transmetim në DCH & FACH të ndara.

- ***Transmetim në DCH për volum të lartë trafiku***

Në konsideratë trafiku dhe kohëzuesit e pasivitetit, modulimi, shpejtësia e të dhënave për kanal, për tip UE mobile, tipi rrjetit R99, R4 etj. **Konsiderata 1**: Trafik i lartë të dhënash - transmetim DCH -> FACH normal:

Vol trafiku	*Kapaciteti DCH*	*T1 (s)*	*Kapac FACH*	*T2 (s)*	*Energjia*
1 Mbit	***Dedikuar** /~ 3 Mbps*	*1.5*	*x*	*2*	*1.961 J*
100 Kbit	*Dedikuar /~ 3 Mbps*	*1.5*	*x*	*2*	*1.753 J*
1 Mbit	***Share** / ~ 300 Kbps*	*1.5*	*x*	*2*	*4.061 J*
100 Kbit	*Share / ~ 300 Kbps*	*1.5*	*x*	*2*	*1.961 J*

Tabela 5.8: Transmetimi në DCH per kohezues 1.5 e 2 sek

Kostoja e energjisë së transmetimit në DCH:

$$E1 = \boldsymbol{P}DCH(T1 + t) + \boldsymbol{P}FACH\ T2 + ngritja/zbritja$$

(t −koha e shpenzuar në DCH)

Për më shumë në rastin e parë:

(0.33 s + 1.5 s) * 600 mW + 2s*400 mW = ~ 1.898 J (+ ngr/zbr)

Shikohet se një kanal i dedikuar dhe i ndarë me të tjerë (share) ndikon ndjeshëm në kohën dhe konsumin e shpejtë të energjisë.

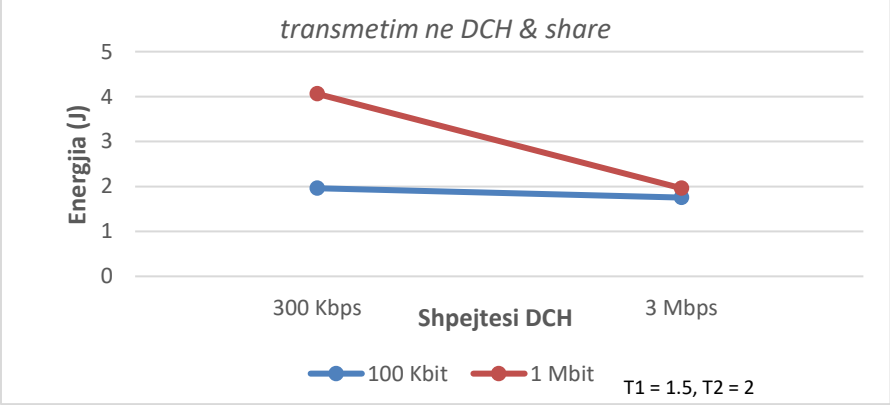


Figura 5.36: Energjia e transmetimit në DCH (1.5 e 2 sek)

- ***Transmetimi ne DCH për volum te ulët trafiku***

Nga vëzhgimi nga matjet dërgimi për të dhëna me shpejtësi të ulta në DCH, rezulton më energji konsumues. Për të njëjtën shpejtësi por madhësi të ndryshme të dhënash shikohet pothuajse e njëjta E e shpenzuar.

Vol trafiku	Kapaciteti DCH	T1 (s)	Kapac FACH	T2 (s)	Energjia
100 Kbit	Ded / ~ 3 Mbps	2	x	4	3.023 J
20 Kbit	Ded / ~ 3 Mbps	2	x	4	3.010 J
100 Kbit	Share / ~ 300 Kbps	2	x	4	3.231 j
20 Kbit	Share / ~ 300 Kbps	2	x	4	3.046 J

Tabela 5.9: Transmetimi në DCH për kohëzues 2 e 4 sek

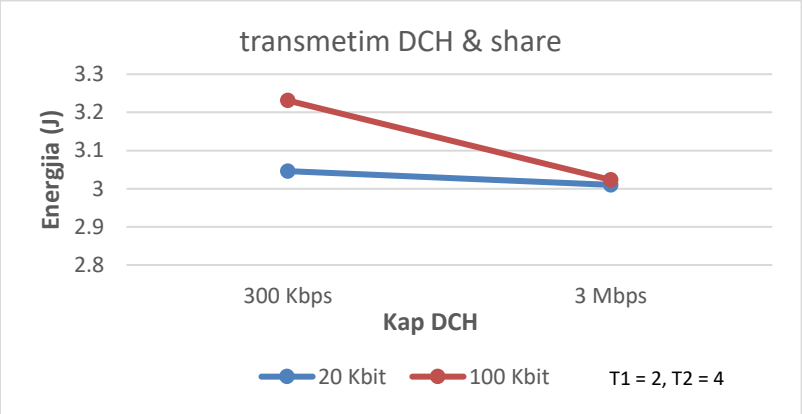


Figura 5.37: Energjia e transmetimit në DCH (2 e 4 sek)

- ***Transmetimi ne FACH për volum të ulët trafiku***

Pra dërgimi i të dhënave vetëm në kanale trafiku FACH për rastet e të dhënave të vogla per nga permasa rezulton me energji ekonomik, krahasuar me grafikun e mëparshëm.

Vol trafiku	Kapac DCH	T1 (s)	Kapac FACH	T2 (s)	Energji
100 Kbit	x	x	32 Kbps	4	2920 J
20 Kbit	x	x	32 Kbps	4	1850 J
100 Kbit	x	x	16 Kbps	4	4000 J
20 Kbit	x	x	16 Kbps	4	1920 J

Tabela 5.10: Transmetimi ne kanal FACH

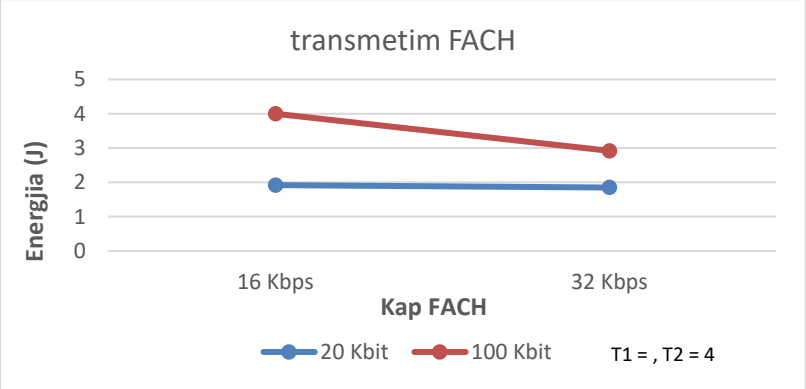


Figura 5.38: Transmetimi ne FACH

Duke ju referuar vlerësimeve energjitike për trafik të lartë e të ulët arrijmë në konkluzionin se:

- Dërgimi data në DCH konsideron dhe shpenzimet ekstra të E në FACH (si rrjedhojë e kohëve të pasivitetit, standart i komunikimit).

- Tranzitime nga FACH <- > DCH <-> IDLE shkaktojnë kosto shtesë E që nuk duhet neglizhuar.

- Dërgimi data për trafik volum të lartë në DCH është me E ekonomik për shkak të shpejtësive e kohëve të transferimit.

- Dërgimi data për trafik volum të ulët e të shpeshtë në DCH rezulton konsumues i lartë i E.

- Për dërgimi në FACH për trafik të ulët është më ekonomik.

Si përfundim mund të themi se :

“ Transferime data me kushte për reduktim tranzitimesh e me vonesa, ndihmon në reduktimin e konsumit të shpejtë të E “.

5.11 Metodat e hulumtimit

Shkenca e vlerësimit të energjisë është një disiplinë e cila ka trashëguar metoda të ndryshme kërkimore. Për shembull, metoda teorike (bazuar në logjikë dhe matematikë), si dhe metodat eksperimentale dhe simulimet janë të kombinuara me një qasje inxhinierike të vlerësimit, matjeve dhe krahasimit. Në këtë seksion, ne shkurtimisht përmbledhim metodën e përdorur. Konsumi i energjisë është rezultat i disa fenomeneve komplekse, duke përfshirë software të ndryshëm si dhe objekte fizike (HW). Ne ndjekim metodën eksperimentale dhe kryejmë eksperimente duke përdorur matje fizike të energjisë, si dhe studimin e literaturave të punimeve të tjera të mëparshme për të kuptuar aspektet kryesore që influencojnë konsumin e energjisë.

Për të marrë dëshmi të përdorimit të zgjidhjes tonë ne e aplikojmë atë në kontekstin e aplikacioneve mobile. Ne përdorim të dy metodat: matjet fizike të energjisë dhe aplikimet sikur në tabelën 7.6 më poshtë në këtë punim, për të analizuar energjinë e komunikimit të aplikimeve me ose pa

zgjidhjen “middleware”. Metodat eksperimentale janë paraqitur në kapitullin e 7 duke përshkruar në detaje mjedisin, si dhe inputet/hyrjet e punësuara dhe matjen në dalje. Më në fund, ne do të “punësojmë” disa krahasime midis një UE pa dhe me zgjidhjen e mesme SW (middleware). Sigurisht si cdo propozim për zgjidhje problematikash me energjinë natyrisht mbeten ende vend për përmirësime të mëtejshme për të cilat do të duhet një fokus në të ardhmen.

Studime në zgjidhjet SW middleware i kanë aplikuar shumë autorë në këtë fushë ku midis tyre mund të ripërmendim si Anannd et al [89], Deng et al [51], si dhe Vergara et al [51] apo the Tailender. Duhet të theksojmë se të gjithë këta autorë pas një studimi të hollësishëm të nuk shpjegojnë në detaje (pjesë nga scriptet apo realizimi i zgjidhjes SW) realizimin SW të zgjidhjes së propozuar në më shumë detaje por vetëm grup testimesh me dhe pa zgjidhjen SW.

Duhet theksuar se dhe vendorët e smartphoneve apo dhe prodhuesit e pajisjeve të rrjetave mobile (HW e SW) i kanë në konsideratë shumë prej këtyre studimeve si dhe shumë prej tyre i implementojnë në SW të rinj (update) si ato RNC psh. Të njëjtën gjë bëjnë dhe prodhuesit e smartphonëve të cilët në versione të reja të Android i marrin në konsideratë këto ndryshime. Megjithatë update të ndryshme si në RNC SW apo dhe UE SW, mund të sjellin ndryshime thelbësore në funksionalitete apo procese e nënprocese të komunikimeve fund më fund pra UE – Rjeti / RNC, c’ka do sillte një rishikim të zgjidhjeve të propozuara.

Qëllimi ynë është të tregojmë që me propozimet e bëra si dhe testimiet e kryera ka vend për rishikim apo optimizim të mënyrës së transferimit të të dhënave. Kjo duhet parë më pas në këndvështrim më të gjerë sepse kompleksiteti i komunikimeve (parametrat, SW’t, HW të ndryshëm etj) krijojnë shumë devijime nga qëllimi ynë.

KAPITULLI VI

6 KONSIDERATA PËR ALGORITMIN

6.1 Algoritmet për eficencë energjie dhe punimet në këtë fushë

Si një objektiv, rizgjedhja e të gjithë kërkesave të rrjetit duke përdorur minimumin e energjisë pa prekur eksperiencën e përdoruesit është çelësi për ruajtjen e energjisë në pajisjet mobile smartphone. Ekzistojnë dy tipe aktivitetesh sikurse i kemi përmendur: *sfond* (background) dhe *aktive* (foreground). Sikurse dihet aplikimet/shërbimet në sfond janë tolerante ndaj vonesave në lidhje me kohën dhe eksperiencën e përdoruesit dhe ato lejojnë disa sekonda vonesë kur fillojnë një sesion. Kjo nuk është e njëjtë me aplikimet aktive (foreground) ku ka një aktivitet të drejtëpërdrejtë të përdoruesit me pajisjen smartphone. Aplikime të tilla si foreground mund të përkufizohen si:

a) *Të rastit*, ku shpërndarja e kërkesave (përdorimi) është e rastit,

b) *Normale*, ku shpërndarja ndjek një ligj fuqie.

Algoritmet e zgjedhjes së transferimeve mund të klasifikohen më së miri në dy klasa: a) *offline* dhe b) *online*, ku ato online mësojnë kohët në kohë reale dhe vendosin variablin e vonesave në kohë bazuar në aktivitetin e aplikimeve. Kjo ndihmon në rastet kur update-t në kohëzuesit e RRC janë bërë në mënyrë të shpeshtë dhe përdoruesi nuk është në dijeni të tyre.

6.2 Algoritmi i zgjedhjes së transmetimeve bazuar në disa kushte

Në këtë pjesë do të diskutojmë e prezantojmë një algoritëm (offline) optimizues i cili tenton të kontrollojë disa parametra dhe disa nga aktivitetet më të rëndësishme të përdoruesit për të reduktuar fuqinë e konsumuar dhe pse jo sinjalizimin në një mënyrë automatike duke mos kompromentuar eksperiencën e pritur nga përdoruesi për atë lloj shërbimi. Algoritmi merr në konsideratë vendosjen e kushteve limit (në lidhje me baterinë, kohën) bazuar në shumë matje të mëparshme të smartphoneve, analizave të matjeve, reagimeve dhe gjithashtu bazuar në impaktin total dhe kontributin e fuqisë së konsumuar. Pjesë të algoritmit do të jenë baza për një zgjidhje të "ruajtjes së energjisë". Ndërkohë një pjesë e tyre mund të jetë marrë në konsideratë prej shumë vendorëve smartphone në release-at e reja të OS të tyre.

Përmirësime në protokollet RRC e RLC janë thelbi në zgjidhjet SW me të cilat mund të arrihen përmirësime të dukshme. Për këtë, algoritmet për zbulimin e tranzitimit dhe nxjerrjes së kohëzuesve T_1 dhe T_2 janë një ide e mirë që mund të përdoret në algoritmin tonë në faza të mëvonshme të tij. P.sh tentimi i një pakete *ping* nga UE me përmasa të ndryshme të paketave dhe monitorimi në të njëjtën kohë në UE si dhe matja e timeout-it dhe e madhësisë së paketave të cilat bëjnë diferencën. Gjithashtu, përdorimi i opsioneve të pritjeve (queing) për transferime të shumëfishta njëkohësisht është përfituese për ruajtjen e fuqisë. Megjithatë, sfidë është si të konfigurohen kohëzuesit e joaktivitetit T_1 e T_2 në mënyrë automatike e jo manuale. Një kohëzgjatje e shkurtër do të shkaktojë rilidhje të shpeshta të UE, problem ky që duhet të shmanget. Edhe pse në gjendjen Idle nuk ka kosto të energjisë, tranzitimi nga IDLE në DCH shkakton konsumim fuqie

që nuk neglizhohet. Rilidhjet e shpeshta do të shkaktonin problem të madh me vonesat e transmetimit për përdoruesin fundor. Nga ana tjetër, rritja e kohëzgjatjes të kohëzuesve të joaktivitetit do ta mbante UE në një gjendje të panevojshme, do të ulte eficencën e përdorimit të burimeve radio dhe do të lejonte konsum më të lartë energjie. Si përfundim protokolli RRC lejon UE-në të rrisë eficencën e tij dhe të ekonomizojë konsumimin e enegjisë, nën kontrollin nga dy parametrat e rëndësishëm: kohëzuesit të joaktivitetit T_1 e T_2. Konfigurimi i këtyre parametrave është i rëndësishëm, sepse vendosja jo e rregullt e kohës së kohëzuesve do të degradojë eficencën e UE-s dhe do të shkaktojë konsum të panevojshëm energjie.

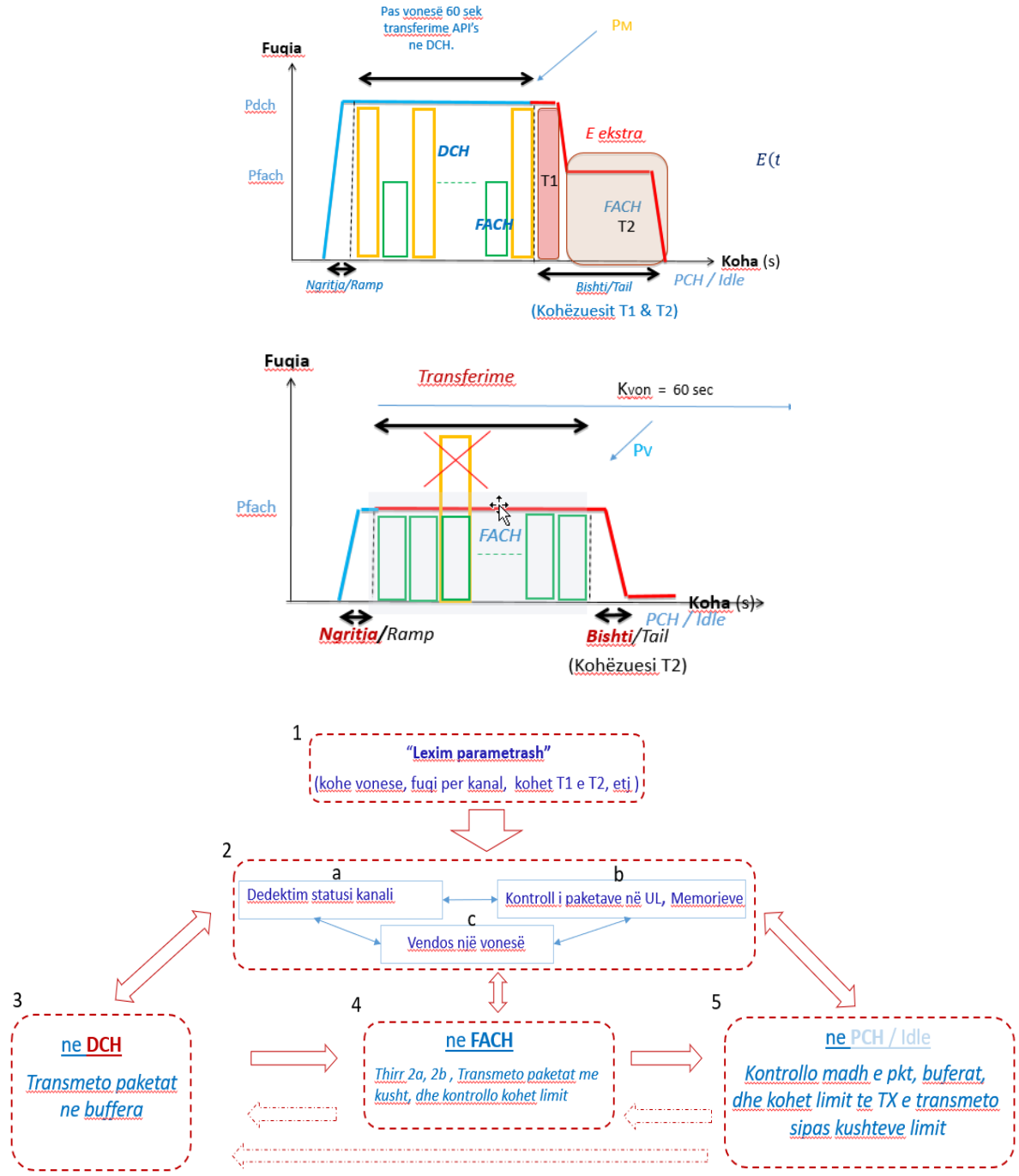


Figura 6.1: Transferimi në DCH e në FACH me vonesë, llogjika

Marrja në konsiderate e nivelit të baterisë, aktivitetit të përdoruesit, kohës së aktivitetit të përdoruesit, kyçjes inteligjente midis ndërfaqes së rrjetit GSM / 3G / WiFi, transmetimeve paralele të të dhënave ose gjatë një thirrjeje aktive zëri si dhe përzgjedhjes së transmetimit në tip kanali PCH/FACH/DCH janë thelbi për algoritmin kompleks, të prezantuar në hapa të përgjithshëm (jo SW në detaje). Algoritmi si një propozim kompleks është tashmë në shumë implementime reale të marra në konsideratë nga shumë prodhues smartphone si dhe sisteme operative bazë Android të personalizuar për model smartphon-i. Kombinimi apo aplikimi patch SW i tij bëhet i vështirë pasi kërkon një njohje të gjerë të strukturës së OS SW e HW të UE. Në zgjidhjen e implementuar SW ne përqendrohemi te pjesa e përzgjedhjes së transmetimeve të të dhënave me bazë zgjedhje transmetimi për kanale në sistemin 3G / 4G duke marrë si të njohura disa parametra të rrjetit. Mëposhtë jepet përshkrim i hapave të një algoritmi kompleks për ruajtje të energjisë në telefonat smartphone në teknologjitë 3G/4G. Gjithashtu në figurën 6.2 jepet grafikisht një përshkrim i shkurtër i punës së algoritmit.

```
 "Powerlevel Based Scheduling transmission Algorithm on Smartphones UE"

 # include function PBSL

// Modul 0:(UE Network IFs & WiFi Status)
{
M0.a) Set UE state in 3G/HSPA+ & 2G/EDGE                          //Dual mode - pa kufizime
       if M1.a.2) met goto M5.a
         else goto M1.a.1
           if M1.a.1 met goto M0.b
             else goto M1.c
           end if
       end if
M0.b) Set to 2G/GSM with EDGE/GPRS on [GSM data Mode]             // shpejtesi e ulet data (pak konsum energjie)
       if M1.a.2 met goto  M0.a
         else goto M1.a.1
       end if
M0.c) Set 2G with no EDGE/GPRS & scren ON (with reduced backlight)  // [PSM status]
       if M5.a are met goto M1.a
         else goto M0.a
       end if
M0.d) Set UE in PSM (M0.c & screen OFF & IF's OFF except 2G)      // PSM mode
       if M1.c is meet goto M0.d
         else goto M1.b
       end if
M0.e) Scanning for Wi-Fi available & connect
      if "connected" goto M1.a & M6.a.1
         else goto M0.e.2                                       // transferon gjithe paketat ne Pm & Pv
   M0.e.1) check Time Tw of repetition (apps)                   // periodicitetit apl
   M0.e.2) switch IF OFF after 3 sec of data inactivity
               else M0.e
      end if
M0.f) Set screen OFF if M4.a.2 met
         else screen ON
         end if
     end if
}
```

```
38
39  // Modul 1: (Decide based on Battery status (L1 & L2))
40  {
41  M1.a) Check (BattL) every 1 hour report BL (L1 & L2)            // statusi i batterise cdo 1 ore
42         if M1.d met goto M6.a.1 (transfer Pm & Pv)
43           if M1.c met goto M0.c
44             if M1.a.1 is met goto M0.b
45               else goto M1.a.2
46             end if
47           end if
48         end if
49     M1.a.1) if BL≤ 35% goto M1.b & M0.b                           // nese poshte 35%
50              else goto M1.a.2
51             end if
52     M1.a.2) if BL ≥ 35% & M5.a.2 meet goto M0.a, M0.e & M4.d     // nese mbi 35%
53              else goto M1.a.1
54             end if
55  M1.b) Check/store BL every 5 minutes till M1.c is met
56             else goto M0.b.
57  M1.c) if BL ≤ 15% goto M0.c or M0.d) and alarm for charging till M1.a.2 is meet    // nese <15%
58             else goto M1.a.2
59  M1.d) if charging mode goto M0.a & M0.e & transfer Pm & Pv         // nese ne karrikim
60             else goto M1.a
61        end if
62      end if
63  }
64
65  // Modul 2:(Decide based on the 24h Time, User activity)          // ne lidhje me kohen e aktivitetit
66  {
67  M2.a) for T24 € [23:59 -- 06:30] goto M1.a.1                     // 23:59--06:30 kohe pushimi
68             else goto M2.a.1
69            end for
70     M2.a.1) for T24 € [23:59 -- 06:30] and M1.a.2 conditions are meet goto M0.b
71             else goto M2.a
72            end for
73  M2.b) for T24 € [06:30 -- 23:59] & M5.a.2 is met goto M1.a.1 if OK goto M4.b.1
74           else goto M2.b.1
75          end for
76     M2.b.1) for T24 between [06:30 -- 23:59] (not T1) & M5.a.1 are met goto M1.a.2
77            else goto M2.b
78          end for
79  M2.d) change T24 based on preference.
80      end for
81  }
82
83  // Moduli 3:(Decide based on the UE screen status - LCD only)    // nese ekrani eshte i ndezur ka
84  {                                                               // aktivitet perdoruesi
85  M3.a) if screen is OFF goto M2.a
86        else goto M2.b
87       end if
88  M3.b) If screen is ON goto M1.a.1 and check for M4.a
89        else goto M4.b
90       end if
91  M3.c) if screen is ON → no activity for 60 sec goto lock screen & M0.d
92        else goto M3.b
93       end if
94  }
95
96  // Module 4:(Decide based on the user activity & Paralell CS or PS call)
97  {
98  M4.a)If active CS goto M1.a.1 & M4.b
99        else goto M0.b
100    M4.a.1) if M4.b meets and
101             else goto 4.b.1
102    M4.a.2) if no active voice (CS) calls goto M0.f & M4.b
103                else
104 M4.b) if active PS calls (UL or DL PS data) goto M5.a.1
105       else goto M6.a.1                              //(transfero te dhenat e aplikimeve te vena ne pritje)
```

```
       end if
       if new adeded applications goto M4.c
M4.c) Call and store if M4.b are meet
        else goto M4.d
M4.d) Discover applications periodic time and Set periodic transfer to one of 2 options:
      M4.d.1) call M3.b and if meet goto M4.b meets.
      M4.d.2) Application max periodic time seen if new applications are added.
        end if
      end if
{

// Modul 5:(Decide based on the user RSSI values)
(RA, Rbuf, RSSI,)
{
M5.a) Measure RSSI (Ra) & store (Rbuf) every 1 sec and on 5th sec do average of Ra.
   M5.a.1) if Ra ≤ -90 RSSI
            stop transfer background (planned) and set queued (M6.a.2) transfers or M0.c till M5.a.2 is met
            else goto M5.a.2.
           end if
   M5.a.2) if -90 ≤ Ra ≤ -70 is meet goto M1.a.1 and M4.b.1    // (per Ra ≤-70 dhe ato Ra ≥ -70 jane perfshire)
            else goto M1.a
          end if
}


// Modul 6: (Algoritmi i perzgjedhjes se transferimeve data ne drejtimin UL )

(B_PCH-DCH, B_FACH-DCH, Kpastr)
M6.a)
{
1: for each new packet p do
2:    if p:size ≥ B_FACH-DCH then
3:      Pm <--p sorted by shortest deadline
4:    else
5:      Pv <--p sorted by shortest deadline
6:    end if
7: end for

//UE perzgjedhje transmetimi baze gjendjeje. UE fillimisht ne PCH.
8: while Pm U Pv ≠ Ø do

M6.a.1)
9:     if UE state = DCH then
10:       transmit Pm
11:       transmit Pv
12:    end if

M6.a.2)
13:    if UE state = FACH then
14:       sum ← 0
15:       while Pv ≠ Ø do
16:         p ← Pv.front
17:         sum ← p.size + sum
18:         if sum > B_FACH-DCH then
19:            wait Kpastr
20:            sum ← p.size
21:         end if
22:         transmit p
23:       end while
24:    end if

M6.a.3)
25:   if UE state = PCH then
26:      Kafert ← shortest deadline €(Pm U Pv)     // pret për kohën më të afërt Kafert për të dërguar packetat
27:      wait until Kafert
28:      choose p € Pm U Pv with Kpastr
29:      if p € Pm then
30:         transmit Pm                              //Con ne DCH
31:      else                                        //Paketat me kohen me te shkurter limit ne Pv
32:       compute VL ← Σp.size ∀ p € Pm
33:       if VL ≥ B PCH-DCH then
34:          transmit Pm                           //Con ne DCH nese plotesohet kushti 33
35:       else                                     //Zgjidh kanalin bazuar ne Pv
36:         compute VS ← Σp.size ∀ p € Pv
```

```
37:         ch ← ChannelChoice(Vs)                  //Function 1
38:         if ch = DCH then
39:            transmit Pv                          //Con ne DCH
40:            transmit Pm
41:         else
42:            transmit Pm                          //Con ne FACH
43:            wait Kapstr
44:        end if
45:      end if
46:     end if
47:  end if
48: end while

Funksion 1 - ChannelChoice
Input: Vs                              //Volumi i Pv ne bytes
Output: channel
     P1; P2                            //Konsumi i fuqise ne DCH dhe FACH
     T1, Kpastr; B_FACH-DCH
  if Vs < P1/ P2 · B_FACH-DCH· T1/Kpastr then
        channel ← FACH
  else
        channel ← DCH
  end if
}


// Modul 7: (State Promotion and Demotions)
(P1, P2, D1, D2 RTT,)
//State Promotion P1: IDLE→FACH→DCH, P2: IDLE→DCH
{
M7.a)UE state in PCH                                    // UE ne kanal PCH, dergon paketa per ndryshim kanali
    M7.a.1) UE send min bytes. a Server echoes min bytes (compute/save RTT)  // = FACH
    M7.a.2) UE send max bytes. a Server echoes min bytes (compute/save RTT)  // = DCH
    M7.a.3) UE record RTT for step 3
M7.b) Report P1 if Δt >> normal RTT. Otherwise report P2
//State Demotion D1 (T1): DCH → FACH, D2 (T1+T2): FACH → PCH (IDLE). UDP packet are sent
M7.c) For n =0 to 10 do
    M7.c.1) UE send max bytes. Server echoes min bytes (compute/save RTT) → switch to DCH → UE sleeps for n sec
    M7.c.2) UE send min bytes. Server echoes min bytes (compute/save RTT) → UE record RTT Δt1 for step M7.c.2
  end for
M.7.d) For n =0 to 10 do
    M.7.d.1) UE send max bytes.
        Server echoes min bytes (compute/save RTT) → switch to DCH → UE sleeps for n sec
    M7.d.2) UE sends max bytes.
        Server echoes min bytes (compute/save RTT)→ UE record RTT Δt2 for step M7.d.2
M7.e) Report D1 if Δt1 and Δt2 are the same,
        else report D2
M7.f) repeat daily M7 and store
   end for
}

// Modul 8: (Infer RLC Buffer Threshold)
 (T1, T2, RTT, MaxBytes, MinBytes, MinDiff, LargeIncr, Small Incr, IPT )
//State is PCH (PCH - DCH)
While (MinB - MaxB > MinDiff) do
    Send TestSizePacket = (MaxB - MinB)/2 and wait IPT and compute RTT (for state status))
       if RTT ≈ RTTP1
//State is FACH (FACH - DCH)
   Send TestPacket = trial Size + Incr (largeIncr) & wait IPT and compute RTT (for state status) or
       if RTT ≈ RTTD2 go to next step
   Send TestPacket = trial Size - Incr (smallIncr) & wait IPT and compute RTT (for state status)
       Wait for T2+T1 (DCH to PCH) and Do
   Send TestPacket = trial Size + Incr (smallIncr) & wait IPT and compute RTT (for state status)
        Store buff_TestPacket
end for
}
```

```
// Modul 9: (reject background transfer during BH when RTT is >> normal RTT in DCH)

M9.a) if time € [10:00 -- 11:00] & [13:00 -- 14:00] & [21:00 -- 23:00]
         send a testpacket & wait IPT and compute RTT (trial for PCH to DCH)
            if RTT >> ave RTT skip ongoing background (QL or QS) and add in ques
               else goto M1.a & M6.a
            end if

// Modul 10: Shutdown
{
    if unload required
M10.a) Deactivate conditions [No restrictions]
      else go to M1.a
    end if
  }
}
```

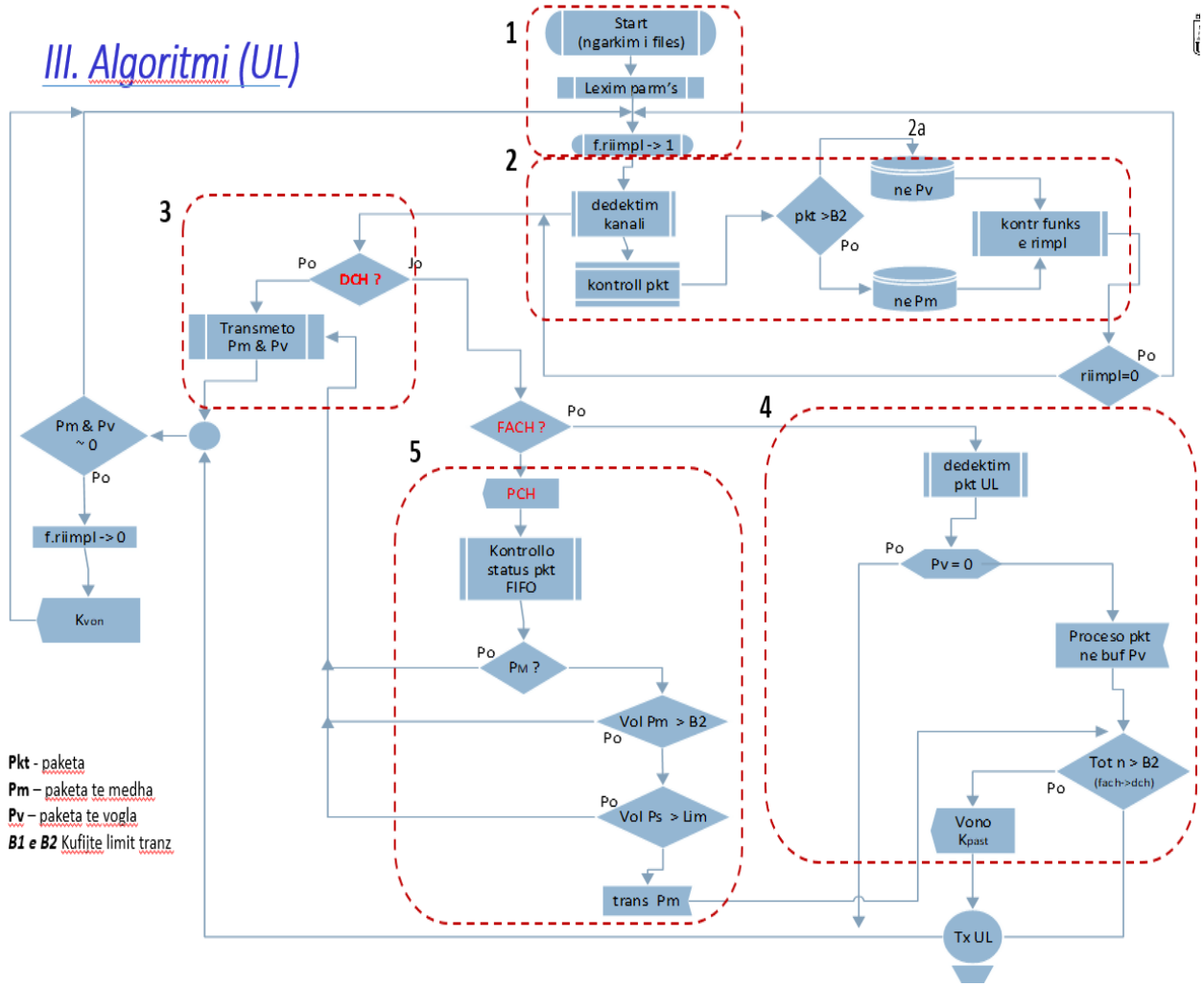


Figura 6.2.a: Algoritmi për ruajtjen e energjisë

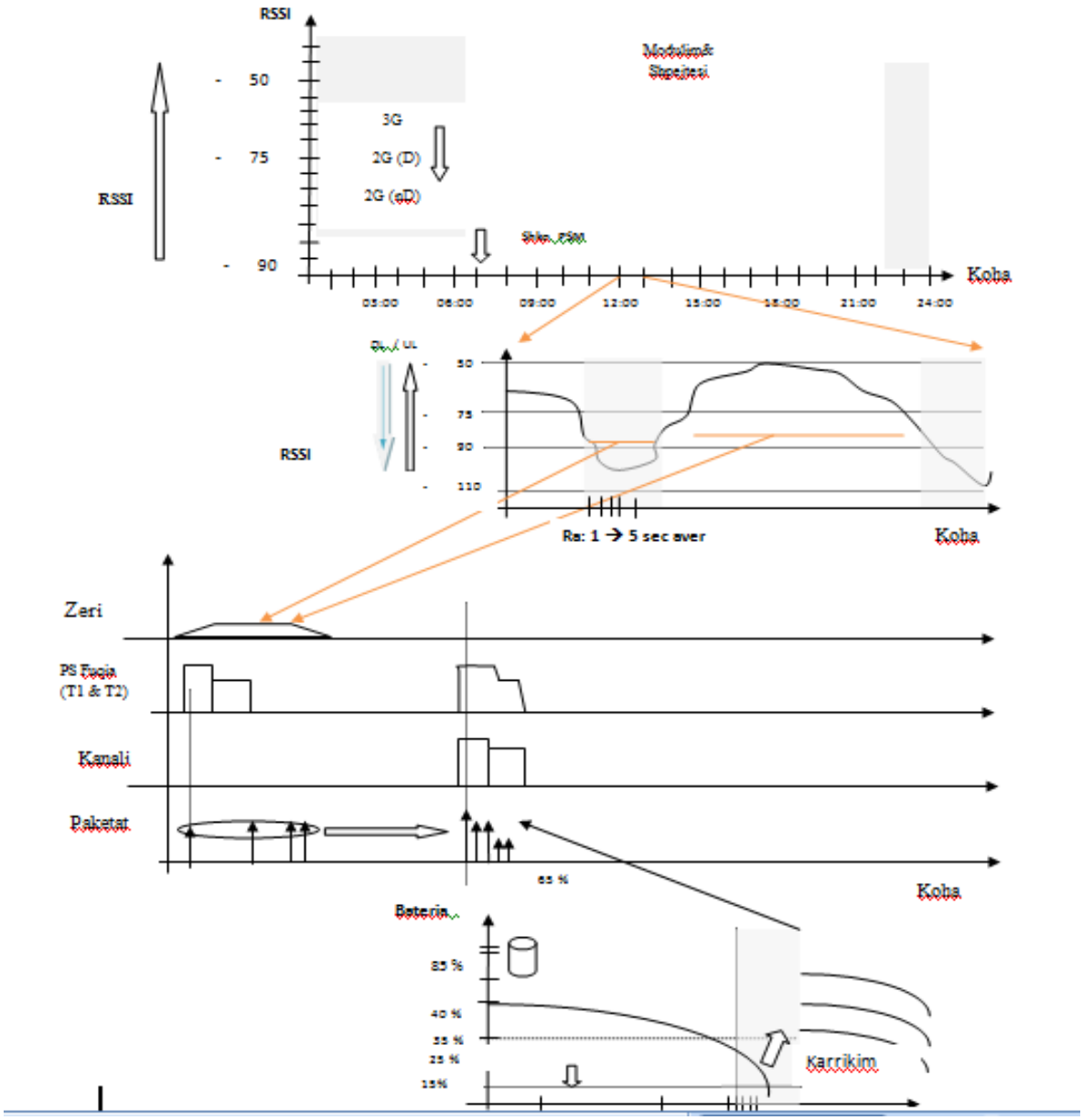


Figura 6.2.b: Konsiderata për përmirësimin energjitik në UE

Aplikimet gjenerojnë trafik me kërkesa të ndryshme (p.sh., trafiku i ndjeshëm nga vonesat ose best effort) dhe kështu afrimi për të reduktuar potencialisht konsumin e energjisë do të duhet t'i marrin në konsideratë këto kërkesa. Ne argumentojnë, se zhvilluesit/programuesit mund të kategorizojnë transmetimet e ndryshme të aplikimeve të tyre bazuar në kritere të tilla si kërkesat e aplikimit ose karakteristikat/feature themelore dhe të ulin prioritetet për komunikimin më pak të rëndësishëm, por ende të nevojshëm në terma të QoS. Pastaj, klasat e trafikut me origjinë nga aplikime të ndryshme në mënyrë efikase mund të përshtaten (psh, nga sistemi operativ) për të krijuar një trafik energji-efiçent dhe për të zvogëluar E(S) referuar figurës 5.9.

Trafiku sfond (background) është kategorizuar shpesh brenda klasës së trafikut background përcaktuar nga QoS e standardit UMTS apo 3G. Klasa e trafikut të sfondit të UMTS-it apo LTEs karakterizohet nga kërkesa me bandwidth të ulët, transmetim të dhënash me ndërprerje dhe

asimetrike (UL / DL) dhe një vonesë të lejueshme derisa destinacioni nuk pret të dhëna brenda një kohe të caktuar. Mungesa kohore apo e performancës së kërkesave të rrepta krijon një mundësi për të eksploruar një kompromis të performancës së energjisë. Por një pjesë të konsderueshme zënë dhe analizmi i trafikut të të dhënave si dhe parë nga këndvështimi i kanaleve dhe madhësisë së paketave si dhe protokollit të përdorur.

Në shembullin në figurën 6.3 në trafikun origjinal të të dhënave në drejtimin Uplink (UE => Rrjeti/RNC), ku në një test të disa autorëve të përmendur [55] janë dërguar rreth 7 paketa me madhësi prej 100 B secila për çdo 300 milisekonda. Kur këto paketa agregohen dhe të dërguara në grup kjo çon në një konsum 42 % më të lartë të energjisë. Në mënyrë që të planifikojmë trafikun në sfond në mënyrë efiçent-energjie, njohja e parametrave të shtresës radio bëhet e domosdoshme. Figura 6.3 tregon një shembull të kësaj ngjarje, ku paketa të vogla të të dhënave nga një gjurmim (trace) real janë të agreguara dhe të dërguara në një burst duke shkaktuar një tranzicion të gjendjeve drejt DCH (pasi është tejkaluar kufiri i bufferit B_2 / PM) dhe duke çuar në më shumë konsum të energjisë se sa në rastin e dërgimit të tyre në kanalin FACH.

Shumë prej aplikimeve si dhe algoritmeve të propozuar që kryejnë agregimin e të dhënave, nuk janë të vetëdijshëm në lidhje me parametrat e shtresës radio të tilla si kufijtë limit të bufferave të të dhënave RLC. Natyrisht, kjo nuk është optimale në kuptimin që këto qasje do të agregojnë transfertat e vogla të të dhënave duke shkaktuar tranzicione të gjendjeve UE te gjendja që konsumon më shumë kur nuk nevojitet.

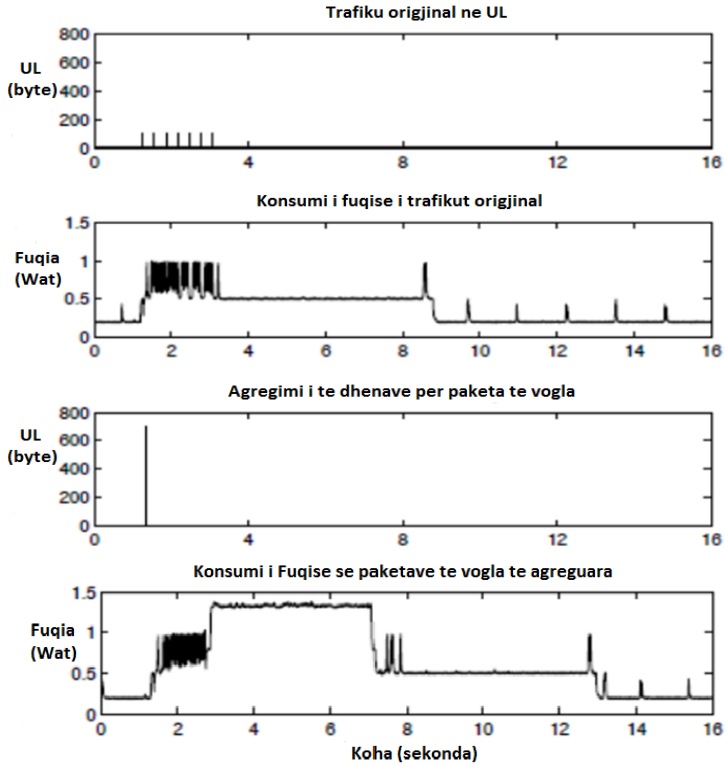


Figura 6.3: Shembull i efektit të agregimit të paketave të vogla në konsumin e fuqisë

Siç kemi treguar dhe në kapitujt e mëparshëm, ndikimi i parametrave të rrjetit të tilla si bishtat (tails), të dhënat e memorjeve/buferave RLC ose cilësia e radio lidhjeve mbi konsumin e energjisë e UE është e lartë. Megjithatë, këto parametra nuk janë në dispozicion për shtresën aplikative, pra

të menaxhueshme nga përdoruesi. Prandaj, shumë autorë në këtë fushë kanë zhvilluar algoritme për të siguruar vlerësime të kohëzuesve të tranzicionit të gjendjeve dhe pragjeve të bufferave RLC c'ka do të ndihmonte në kuptimin e sjelljes së rrjetit radio dhe konsumit të energjisë për kanal.

6.3 Metoda dhe funksionimi i planifikimit të transferimeve për kanal

Metoda e përzgjedhjes së transferimeve apo moduli 6 në algoritmin e përbërë, pjesë e propozuar dhe nga autoret në [55] është fokusuar në eficencën e zgjedhjes së transmetimeve të klasës së trafikut data background / sfond në 3G (UMTS) në lidhje me kërkesat për konsumimin e energjisë. Ne e zgjerojme me tej me disa optimizime dhe me teknologjine 4G. Metoda tenton të bëjë që UE të qëndrojë sa më gjatë të jetë e mundur në kanalin PCH e FACH, ku harxhimi i energjisë është minimal dhe më i ulët sesa në DCH për transmetime të vogla. Qëlllimi është që planifikimi i transmetimeve tenton të dërgojë të dhëna në kanalin FACH, e cila është më pak konsumuese sesa kanali DCH në lidhje me energjisë. Metoda do të merret me "ri-organizimin" e trafikut në UL, pasi vetëm ky lloj trafiku është i menaxhueshëm nga UE dhe me mundësi menaxhimi. Në lidhje me drejtimin DL është RNC ajo që më së shumti kontrollon atë (në raste kur nuk ka një kërkesë të mëparshme nga ky UE). Metoda e planifikimit përcakton dy pritje të ndryshme për të zgjedhur paketat në UL në varësi të madhësisë së tyre dhe kjo në këndvështrimin e krahasimit me kufijtë limit të memorjeve RLC. Ka dy memorje për paketat e ndara sipas madhësive të tyre dhe specifikisht të vogla dhe një memorje për paketat e mëdha. Kjo metodë vendos për dy memorje sepse është një kufi limit i cili duhet të merret në konsideratë pasi në saj të tij krijohen dhe tranzitimet e gjendjeve. Këto dy "kufij" janë vetëm për faktin se transmetimi në UL përdor kufirin (limitin) e memories në protokollin RLC (të shtresës së 2-3 në modelin TCP/IP sikurse në figurat mëposhtë) për të krijuar tranzitimin në kanale. Për të kufizuar tranzitimet e shpeshta pra dhe për rrjedhojë konsum energjie në kanale me energji të lartë, paketat duhet të vendosen në memorjen e vogël *Pv* ose të madhe *Pm* sipas kushteve në transmetim dhe kosto energjitike.

Për të gjitha këto do të duhet të vendosim për dy memorje të ndryshme në mënyrë që të dallojmë paketat që do të dërgohen në FACH ose te DCH (apo e njëjta llogjikë për 4G): $P_M(Q_1)$ për paketat e mëdha dhe $P_V(Q_2)$ për paketat e vogla. Çdo paketë e re do të vihet në pritje në varësi të madhësisë së saj. Ato do të klasifikohen bazuar në afatin kohor më të afërt limit. Nëse madhësia e saj është më e madhe sesa B_2 (si në figurën mësipër) ajo futet në pritje në P_M, përndryshe në P_V. Mekanizmi përgjegjës për transmetimin është në gjendje aktive pra për transmetimin, për sa kohë ka paketa në pritje. Duke supozuar se ka paketa egzistuese në pritje, transmetimi do të varet nga gjendja e UE, e cila mund të jetë DCH, FACH ose PCH. Do të pranohet se gjendja/kanali PCH është gjendja bazë nga e cila fillon mekanizmi i transmetimit të të dhënave. Eshtë dhe më lehtësisht ndihmuese për arsye studimi, si dhe gjendja kur një përdorues kalon porsa është i lidhur me bërthmën/core të rrjetit SGSN e GGSN.

Nëse gjendja e UE është në kanalin energjitik të lartë apo DCH, paketat e ruajtura në të dyja pritjet P_M e P_V transmetohen. Kështu që në marrim avantazh nga shpejtësia e të dhënave të gjendjes DCH për ti transmetuar të gjitha paketat si një burst i vetëm, pasi në këtë gjendje në varësi të distancës, kushteve radio të rrjetit si dhe specifikimeve të tjera të rrjetit shpejtësia e transmetimit është e lartë por dhe e variueshme. Nëse UE është në kanalin energjitik të mesëm FACH, paketat e ruajtura në

P_V janë transmetuar një nga një në mënyrë që të shmangim tranzitimin e gjendjeve të UE në DCH, pra limitohet transmetimi që të mos tejkalohet kufiri i bufferit të poshtëm. Kështu që transmetimi i të dhënave me kapacitet të vogël kryhet në kanalin FACH (me plotësimin e kushtit të bufferit) duke reduktuar tranzitimet e panevojshme të UE në DCH. Kjo lloj metodologjie ka treguar se përmirëson konsumin e energjisë *me të paktën 25 - 40%*.

Metodologjia e planifikimit gjatë implementimit në kernel përbëhet nga 2 pjesë apo mekanizma: një pjesë përgjegjëse për vendosjen në pritje të paketave, pra në dy pritje të ndryshme në varësi të madhësive të tyre dhe një pjesë përgjegjëse për transmetimin e tyre që është i bazuar në gjendjen e UE për të transmetuar paketat e pritjeve. Metodologjia filllimisht merr disa variabla si hyrje të cilat janë matur më parë për operatorin specifik nën testim, dhe të cilat duhet të jenë si parametra të njohur apo hyrës për modulin energjitik: B_1 është pragu (kufiri) i memorjes RLC për të kaluar direkt nga kanali PCH në kanalin me gjendje më të lartë energjitike DCH. B_2 është pragu (kufiri) i memorjes RLC për të kaluar nga kanali me gjendje të mesme energjitike FACH në kanalin me gjendje më të lartë energjitike DCH.

Duhet thënë se kalimi nga kanali PCH në FACH ndodh në cdo moment që kemi për të transmetuar data me volum të ulët. *Kp* tregon periudhën e kohës së duhur për të pastruar (fshirë) memorjen RLC pas transmetimit të të dhënave e cila varet nga shpejtësia e të dhënave të kanalit. T_1 është kohëzuesi i pasivitetit për të kaluar UE nga gjendja DCH në FACH. T_2 është kohëzuesi i pasivitetit për të kaluar UE nga kanali FACH në gjendje fjetjeje apo PCH.

6.3.1 Shembull funksionimi i metodës së planifikimit të transmetimeve

Për të treguar se si metodologjia funksionion duhet të simulojmë nëpërmjet një testimi nëpërmjet dërgimit te disa paketave me përmasa fikse për një periudhë kohore nga njëra tjetra. Në shembullin tonë supozojmë se memorja e brendshme e vogël P_V ka 7 paketa me nga 100 byte. UE ndodhet në gjendjen UE aktualisht. Ne vëzhgojmë që duhet të futet në llogjikë një numërues i quajtur shuma për të kontrolluar nëse kalohet madhësia e kufirit B_2.

Figura 6.4 tregon transmetimin e paketave të para në pritje nga shtresat e mësipërme në shtresat e mëposhtëme, pra në drejtimin Uplink (UE drejt RNC). Paketa e parë është marrë nga pritja dhe vlera e madhësisë së saj është shtuar te *shuma n* dhe është kontrolluar nëse tejkalon kufirin e memorjes FACH në DCH. Më pas paketa transmetohet. Procesi është përsëritur kështu me rradhë duke u llogaritur gjithmonë s*huma n* e cila krahasohet me kufijtë limit të memorjeve RLC. Paralelisht kontrollohet për kohët e transmetimit të paketave. Ky proces përseritet vazhdimisht gjatë cdo dërgimi pakete dhe ka një rëndësi thelbesore në gjithë procedurën e ruajtjes së energjisë. Normalisht këto veprime në kernel kërkojnë një ndërveprim shtesë të CPUs sikurse simulohet dhe nga tabelat më poshtë për punë të CPUs dhe për më tej dhe energji të vetën e cila në nivel kerneli është shumë e ulët krahasuar me përfitimin energjitik.

Megjithatë, përpara transmetimit të paketës së gjashtë kufiri i memorjes FACH - DCH që çon në gjendje tranzicioni tejkalohet, kështu që paketa duhet të presë për për kohën e vonesës apo *Kp* përpara se të jetë dërguar për transmetim në mënyrë që të pastrohet kufiri i RLC dhe të mbahet UE në gjendjen FACH. Procesi vazhdon deri sa memorja për paketat e vogla apo $P_V(Q2)$ të jetë bosh.

Marrim në konsideratë rastin kur UE është në gjendjen PCH, ne presim derisa *afati kohor limit (deadline)* më i afërt i ruajtjes së paketave në 2 memorjet e paketave skadon. Pikërisht, përpara se të skadojë *afati (deadline)* më i afërt, algoritmi kërkon nëse afati i caktuar ishte i lidhur me volumet e paketave të mëdha apo paketat e vogla. Nëse afati i caktuar i përket / takon një pakete të madhe, P_M do të transmetohet duke kaluar UE nga PCH në kanalin DCH direkt. Në këtë lloj mënyre menaxhohet sasia apo volumi i paketave në dalje duke mos lejuar tejkalimin e madhësises së bufferit që inicion një tranzitim.

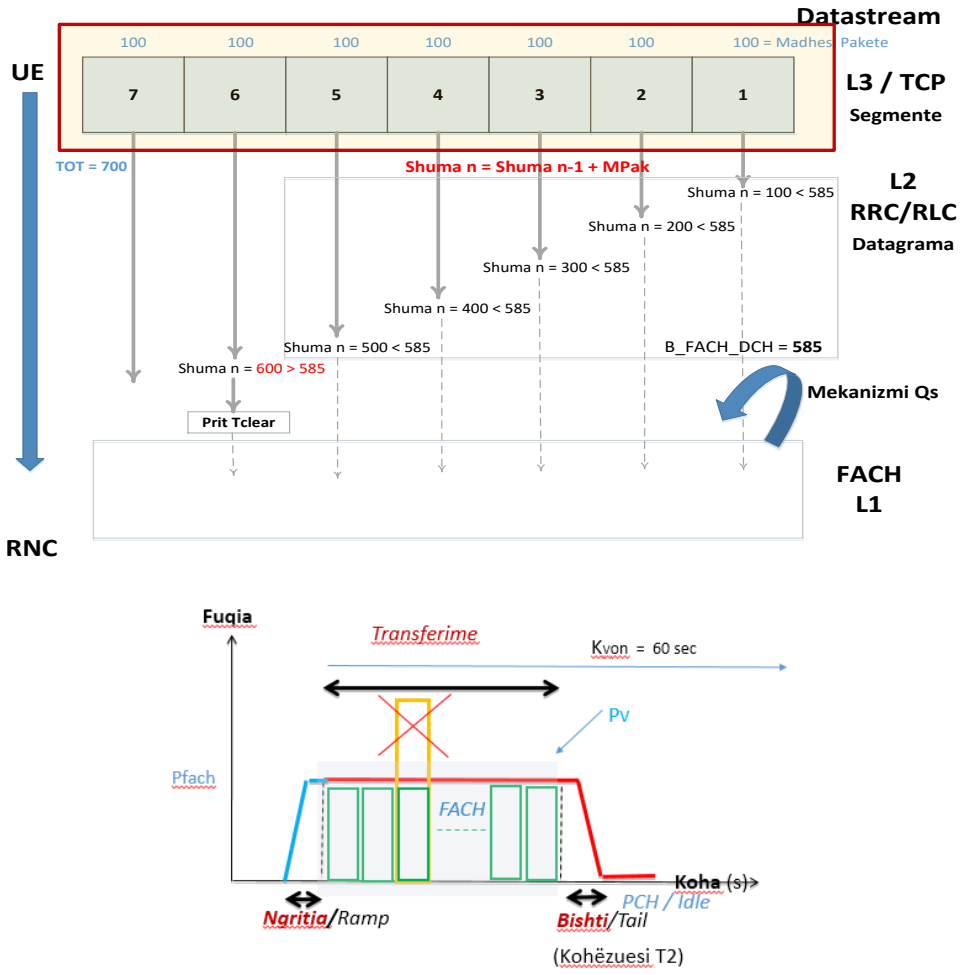


Figura 6.4: Shembulli i transmetimit të UE në kanalin fach 3G

Përndryshe, ne llogarisim madhësinë e të gjithë paketave të ruajtura në P_M, dhe nëse shuma e transmetuar tejkalon vlerën e B_1, P_M *(*Q_2*)* do të transmetohet duke cuar përsëri UE në DCH nga gjendja apo kanali PCH. Nëse nuk e tejkalon, ne llogarisim gjithashtu volumin e P_V (V_v). V_v si vlerë është llogaritur në vlerën fikse nga formula ku disa vlera duhet të jenë të njohura më parë:

$$V_V = P_1/P_2 \cdot B_2 \cdot T_1/K_p \quad (6.1) \quad [55]$$

ku P_1 dhe P_2 janë fuqitë e konsumuara të gjendjeve DCH dhe FACH. Kjo vlerë përcakton nëse transmetimi do të vazhdojë të konsumojë më shumë energji në FCH apo DCH. *Nëse* V_v *tejkalon vlerën, të dyja Pritjet janë transmetuar dhe kjo çon/drejton UE-n të vendoset në DCH. Nëse jo, vetëm përmbajtja e* P_V *është transmetuar duke drejtuar UE në FACH dhe ne presim për kohën Kp.*

Kjo procedurë është ndodh apo do të përsëritet kurdo kur aty ka paketa në pritje, sepse në smartphonet e sotëm gjithmonë ka paketa në transmetim dhe për shkak të llogjikës së mbaj-gjallë. Por sikurse mund të shihet metodologjia kërkon një njohje të disa parametrave (si fuqitë për kanal, kohëzuesit e pasivitetit, vlerat e memorjeve, apo dhe kohët e pritjës së pastrimit të memorjeve etj) të cilat duhen futur automatikisht në modulin SW të OS:

Të gjitha matjet janë kryer duke përdorur një kartë SIM nga operatori shqiptar me nr 0666xx për të siguruar akses të plotë të kapaciteteve të rrjetit në dispozicion 3G. Në 4G janë bërë disa testime të tjera por jo intesivisht si në 3G. Përveç nëse specifikohet ndryshe, të gjitha matjet janë kryer në të njëjtin vend / pozicion fiks ku fuqia e marrjes së sinjalit nuk ndryshon shumë dhe sinjali i marrë është në vlera të mira. Të dhënat e bufferave RLC në uplink dhe downlink janë përdorur për të llogaritur vëllimin e trafikut dhe për të shkaktuar trigerimet e tranzicionit të gjendjeve apo kanaleve më të larta. Kur të dhënat në buffer tejkalojnë pragun, kontrolleri RNC apo MME rivlerëson shpërndarjen e burimeve dhe kërkon ndryshim në gjendjen e UE sikurse kemi përmendur. Ne kemi matur pragjet/kufijtë në drejtimet ngjitje dhe zbritje që shkaktojnë tranzicionet e treguara mëposhtë në këtë punim, duke dërguar paketa UDP të madhësive të ndryshme dhe duke vëzhguar energjinë faktike aktuale dhe duke përdorur setup-in tonë të matjes. Vini re se tranzicioni PCH – DCH (ne 3G) është më i mundshëm të ndodhë kur paketa e dërguar është më afër kufirit të sipërm.

Matjet tona janë kryer në intervale të ndryshëm e në periudha të ndryshme prej vitit 2013 - 2017 dhe jo gjatë BH të operatorit në mënyrë që të shmangim kufizimet në trafik. Ato jane kryer në një pozicion fiks dhe në distancë nga antenat shërbyese me një sinjal jo shumë të fortë (~ 95dbm) si dhe për një operator të ngarkuar ku shpejtësia e të dhënave në DCH (~ disa qindra kbps) nuk është shumë e madhe për shkak të ngarkesës që ka rrjeti.

6.3.2 Përcaktimi i bishtit/Tail

Gjithmonë kjo duhet të nxirret nga nga matje reale në operatorë mobile (nxjerrja e kohëzuesve të pasivitetit). *Për të konkluduar, T_1 dhe T_2 (kohëzuesit e pasivitetit në UL) janë të bazuara në njohuritë që çdo gjendje ka një kohë të vlerësuar të ndryshme vajtje-ardhje (RTT) për një paketë të dhënash.* Vlerësimi i kohëzuesve të pasivitetit mund të bëhet sa herë që një ndryshim i parametrave të rrjetit pritet të ndodhë nga ana e operatorit. Për të llogaritur RTT, ne mund të dërgojmë disa paketa testi me madhësi të paracaktuara nga një UE nën testim. Një server i bën "jehonë" (kthim përgjigje) me një përgjigje duke lejuar llogaritjen e RTT. UE e fillon duke qënë në gjendjen PCH. Për kohëzuesin T_1 të tranzitimit nga DCH, ne shkaktojmë një tranzicion gjendjeje në DCH duke dërguar një paketë provë testi të madhe, e ndjekur nga një sekuencë paketash të vogla. Në një moment, UE do të kalojë poshtë në gjendjen FACH si shkak i madhësive të vogla të paketës. Në këtë kohë, sa më e gjatë RTT, kjo do të jetë tregues i tranzicionit të gjendjes.

Për kohëzuesin e tranzitimit nga FACH apo T_2, mënyra më sipër nuk do të punojë. Me dërgimin e paketave të vogla do të mbajë UE në FACH pafundësisht. Prandaj, duhet të fillojmë me një vlerë të ulët dhe të lartë lidhur me T_2. Duhet thënë se kufijtë e sipërm dhe të poshtëm nuk duhet të jenë në vlera të sakta. Ne fillojmë me hamendësime në ndonjë vlerë dhe në mënyrë të përsëritur të konvergjojmë në T_2 duke zvogëluar hendekun mes tyre. Ne fillojmë me një vlerë kohëzuesi *trial* (të përkohshëm) ndërmjet kufijve. Ne kemi dërguar një paketë provë që shkakton kalimin në

FACH dhe presim për kohëzuesin “trial” dhe rregullojmë kufijtë më të ulët dhe të sipërme të bazuara në gjendjen UE. Nëse UE është në PCH, zgjedhja e kohëzuesit trial është më e gjatë se T_2. Nëse UE është në FACH, kohëzuesi trial është më i shkurtër se T_2. Duke përdorur kërkimin binar dhe një sekuencë të dërgim-pritje-kontroll ne konvergjojnë në një vlerë të saktë të T_2.

Gjithashtu mënyra të ngjashme janë ato me anë të api SW NEP, PowerTutor, GSam apo metoda me matje nëpërmjet analizimt të filave logger në UE si psh ato Android. Kjo pjesë është ngjashmërisht me metodologjinë 1 të matjeve të përmendur më parë.

6.3.3 Përcaktimi i kufirit të bufferit RLC

Për nxjerrjen e kufirit të të dhënave të bufferit RLC që shkaktojnë tranzitim të gjendjeve, siç kemi thënë zakonisht janë katër kufij: dy për kalimin PCH-DCH dhe dy për tranzicionet FACH-DCH (uplink dhe downlink). Për nxjerrjen e kufirit të të dhënave PCH-DCH, ne përdorim një vlerësim të madhësisë se bufferit të shënuar nga një kufi i sipërm dhe i poshtëm respektivisht (MaxBytes dhe MinBytes). Për nxjerrjen e kufirit të FACH-DCH të të dhënave të bufferit, ka mënyra të ndryshme. Në këtë rast, duke shkaktuar më pak tranzitime gjendjesh kursejmë energjinë e përdorur për konkluzion. Në fazën e parë, ne i afrohemi pragut të bufferit duke përdorur “rritës” të mëdhenj të madhësisë së paketës test. Në fazën e dytë ne përdorim rritjet e vogla të paketave (mbi paraardhësen) në mënyrë që të sintonizohemi në vlerësimin e kufirit të bufferit. Madhësia e paketës test është rritur në FACH nga një rritje e madhe (50 bytes) derisa gjendja DCH është shkaktuar.

Psh. merret rasti i tranzitimit në 260 Byte. Për të zbuluar këtë 300 Byte të dërguar në kohë shkaktojnë 5 kalime të gjendjes, prandaj pragu i bufferit është midis 250 dhe 300 byte. *RTT e cila llogaritet për çdo paketë është përdorur për të konkluduar gjendjen aktuale UE*. Pastaj, pas pritjes T_1 për t'u kthyer në FACH nga DCH në pikën kohore 5 (në figurë), rritja e madhësisë është e vendosur për *smallincrement* dhe algoritmi fillon dërgimin e 250 + Byte rritës përsëri (p.sh, 260 bytes në pikën kohore 4 s).

Vëmë re se intervali ndër-paketë duhet të jetë i madh sa për të lejuar pastrimin e bufferit të të dhënave RLC. Shembulli përdor një *InterPacketInterval* të 1 sekondë.

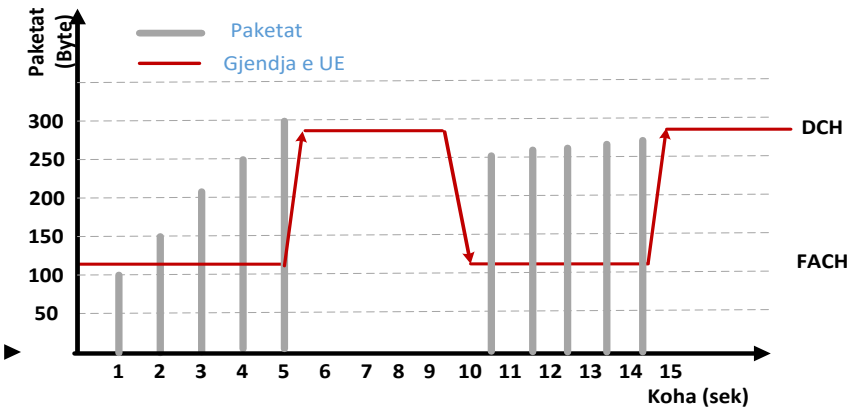


Figure 6.5: Shembulli i nxjerrjes së vlerës së bufferit te UE në UL

Në vazhdim, ne paraqesim metodën për planifikimin uplink të të dhënave që përdor parametrat e nxjerra në seksionet e mëparshëm. Qëllimi i metodës së planifikimit është për të minimizuar konsumin e energjisë me anë të caktimit paketave të vogla të dhënave në gjendjen FACH dhe të shmangë tranzitimet e kushtueshme të gjendjeve në DCH që rrisin shpenzimet e përgjithshme të energjive bisht: bishtin DCH dhe bishtin FACH. Për më shumë, ne do të ndalojmë dërgimin e paketave para afateve të tyre limit, ndërsa ne jemi në gjendjen PCH. Moduli në implementim është i ndarë në dy pjesë: një mekanizëm i orientuar që bën pritjeje paketash dhe që i fut paketat e aplikimit në dy memorje të ndryshme në varësi të madhësisë së tyre, si dhe një mekanizëm të bazuar në gjendjen apo statusin e UE që transmeton paketat aktuale në varësi dhe të kohëve të vonesës.

Le të përcaktojnë problemin si më poshtë. Le të jetë $\mathbf{A_i}$ = {a_1; :::; a_n}grupi i aplikacioneve me kufizime strikte kohore (trafik kohë reale-interaktive) që kryejnë transmetime në kohë reale të të dhënave (p.sh, ndërveprimin e përdoruesit). Le të jete $\mathbf{T_i}$ afati më i shkurtër relativ për çdo paketë të prodhuar nga një aplikim në grupin A_i. Le të jetë $\mathbf{A_b}$ = {b_1; :::; b_n} grupi i aplikacioneve me kufizimet e lejuara në vonesë kohore (trafik background / sfond). Le të jetë $\mathbf{T_b}$ vonesa më e shkurtër e tolerimit për planifikimin e transmetimit për grupin e A_b. Ne përcaktojmë ***Kv*** si kohë maksimale e pritjes/vonesës për një transmetim paketë të origjinuar nga A_b, e cila është koha e nevojshme për kryerjen e transmetimit të të dhënave (të gjitha aplikacionet kanë kohë të tillë maksimale pasi përndryshe vetëm se do të bllokonin trafikun) [55]. Kryesisht në praktikë $\mathbf{T_i < T_b}$, dhe më tej klasifikojmë paketat e grupit të aplikacioneve ($\mathbf{A_i}$ U $\mathbf{A_b}$) në dy kategori në bazë të madhësisë së tyre, M (për të mëdha) dhe V (për të vogla). P_V dhe P_M të janë radhët e pritjeve të paketave të vëna në radhë sipas madhësisë së tyre dhe të renditura duke futur afatin më të shkurtër në krye të radhës. Prioritet është nxjerrja nga matjet e kohëzuesve të pasivitetit T_1 - T_2 dhe pragjet / kufijtë RLC të bufferit B_ PCH_DCH dhe B2 (tranzicionit e gjendjeve respektivisht nga PCH te DCH dhe nga FACH në DCH). ***Kp*** *sikurse kemi thënë është koha për të pastruar nga memorjet RLC e përshkruara në seksionet dhe mund të matet.*

Le të japim hollësi mbi kërkesat kohore për paketat. ***Kv*** paraqet *kërkesat në kohë të trafikut sfond*, p.sh. updatet-e nga RSS, aplikimet Email ose Facebook etj. Këto mund të vendosen nga aplikimet, ose bazuar në kërkesat e energjisë së përdoruesit (profilet e energjisë). $\mathbf{T_f}$ është përcaktuar si *një afat për aplikime të tjera*. Metoda planifikon vetëm transmetimet e të dhënave të sfondit e jo interaktive, dmth, paketa me një kohë maksimale e pritjes dhe që nuk kanë kërkesa të rrepta kohore. ***Kv*** është në praktikë *koha maksimale që algoritmi mund të vonojë një transmetim paketë* dmth, kohën që paketa është origjinuar në aplikim. Le të japim hollësi mbi afatet për paketat. $\mathbf{T_s}$ paraqet afatin limit të UE, p.sh. update-t nga RSS, e-mail ose aplikimet Facebook. $\mathbf{T_i}$ është përcaktuar si afati i shkurtër për aplikime të tjera. Është e qartë se kursimet maksimale të energjisë janë arritur kur këta 2 kohezues $\mathbf{T_i = T_b}$.

Pjesa e parë e Algoritmit te moduli 6 (linja 136-145) i algoritmit të përbërë thjesht vë në rradhë paketat në varësi të madhësisë së tyre të renditura nga koha më e shkurtër që duhet të dërgohet (t) pasi t'i kemi vonuar deri në kohën e tyre të fundit për tu dërguar nga Kv/s. P_V përmban paketa që mund të dërgohen në FACH pa shkaktuar një tranzicion të gjendjes te DCH. Pjesa kryesore e Algoritmit (moduli 6 me rreshtat nga 148 - 191) ka si funksione të transmetojë të dhëna të bazuara

në gjendjen UE nëse ka ndonjë paketë që do të dërgohet. Veprimet që mund të kryejë UE janë: të transmetojë një paketë të vetme, të transmetojë një radhë/queue në grup/burst ose të presë. UE fillon nga gjendja PCH dhe për çdo paketë të dërguar është në gjendje të bëjë dallimin në mes të një tranzicioni të gjendjes të DCH ose FACH me dijenin e kufijve të bufferave RLC dhe kohëzuesve të pasivitet T_1 dhe T_2. Çdo herë që një paketë është dërguar ajo përditëson kohën e transmetimit të fundit dhe gjendjen aktuale. Pas kohëzgjatjes të kohëzuesit të pasivitetit gjendja e UE është ndryshuar. Kur UE është në DCH ne transmetojmë paketat e ruajtura në P_M dhe P_V. Kur UE është FACH, transmeton paketat e ruajtura në P_V duke shmangur shkaktimin e një tranzicioni të kushtueshëm te gjendja DCH, pra kemi një kufizim të shpejtësisë së të dhënave. Kjo është bërë duke dërguar një numër të paketave që shuma e tyre është në më pak bytes sesa B_2. Pastaj, UE pret për kohën K_V para dërgimit të ardhshëm të paketave në pritje. Kjo lejon UE të mbetet në kanalin FACH deri sa memorja P_V të jetë zbrazur. Në rastin kur UE është në PCH, metoda krahason kohën e dërgimit të paketave në krye të P_M dhe P_V dhe pret *për kohën më të afërt për të dërguar* K_{afert}. Pastaj, UE vendos nëse ai do të dërgojë një paketë ose burst që shkakton një tranzicion të gjendjes të DCH ose FACH. Do të dallohen tre raste:

Në fillim, në rast se paketa me kohën më të shkurtër të dërguar është në P_M (linja 129-130), ajo duhet të dërgohet. Të dhënat në P_M do të transmetohen duke shkaktuar një tranzicion te gjendja me nivel më të lartë energjie DCH, dhe kjo sepse kalohet limiti i bufferit të tranzitimit. *Në vazhdim*, në qoftë se paketa është në P_V, UE do të transmetojë paketat në P_M nëse vëllimi në byte i P_M është më i madh sesa kufiri i memorjes PCH - DCH, linja 172-176). Kjo do të shkaktojë një tranzicion te DCH (duhet thënë se kur UE është në kanalin DCH të dyja pritjet do zbrazen pasi nuk ka limitim në këtë gjendje). *Në rastin e tretë*, transmetimi në FACH është i realizueshëm me vëllimin aktual të rradhëve të pritjes. Megjithatë, mund të ndodhi që një vëllim i madh të dhënave në UL në P_V mund të çojë UE për të qëndruar në FACH për një periudhë të gjatë, e cila do të konsumojë më shumë energji se sa dërgimi i radhëve në burst në DCH (pasi shpejtësia e të dhënave në FACH është më e ulët). Zgjedhja e opsionit me më pak konsum, dmth duke dërguar në DCH ose FACH, është kryer nga ana e një funksioni kontrolli të statusit gjendjes që fillon në linjën 193 duke pasur parasysh vëllimin e P_V. Kalimet nga PCH në DCH e anasjelltas përkojnë me llogjikën e komunikimeve në 4G.

Nëse dërgimi në FACH është më pak konsumues, UE do të transmetojë volumin në P_M e cila shkakton një tranzicion te FACH dhe pritet për një kohë K_V. P_V do të zbrazen në gjendjen FACH. Nëse dërgojmë në DCH është më pak konsumuese, UE transmeton të dy rradhët në burst duke shkaktuar një tranzicion te DCH. Funksioni përcaktues për këtë gjë (linjat 193 deri 203) është një optimizim për të shmangur rastin kur një vëllim i madh i të dhënave në P_V (V_S) udhëheq UE për të qëndruar në FACH për një periudhë të gjatë. Duke pasur parasysh konsumin e energjisë të gjendjes DCH dhe FACH (P_{DCH} dhe P_{FACH}) dhe kohën e transmetimit (ϵ) të fluksit të të dhënave në P_V në DCH, kostoja e energjisë së transmetuar në gjendjen DCH është vlerësuar nga ky formulim:

$$E1 = P_{DCH}(T1 + \epsilon) + P_{FACH}\,T2 \quad (6.2)$$

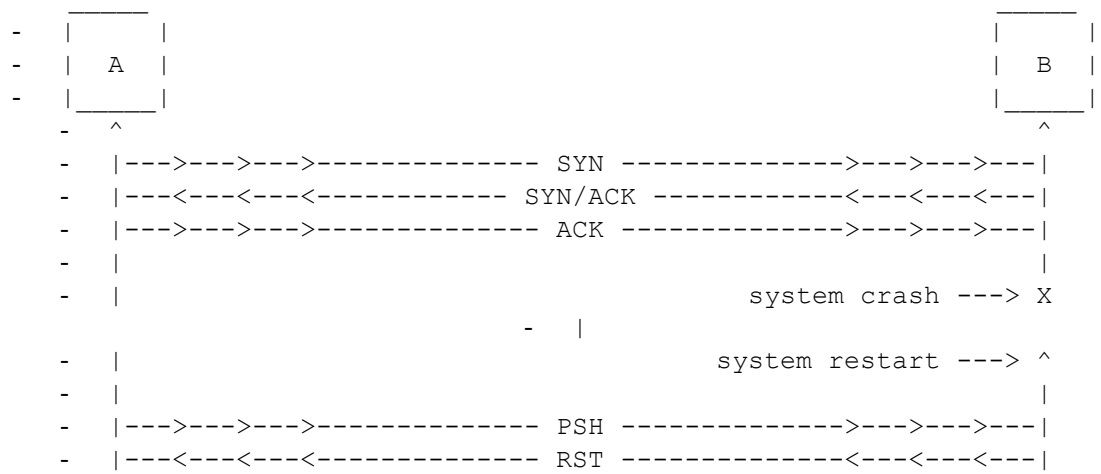
Duke pasur parasysh shkallën e lartë të dhënave të DCH, konsiderojmë se ϵ është i papërfillshëm për vlerësimin. Kostoja e energjisë në gjedjen FACH është vlerësuar nga ky formulim:

$$E2 = P_{FACH}\left(T2 + Kp\frac{Vs}{Q2}\right) \quad (6.3)$$

Prandaj, kur pabarazia E1 > E2 qëndron, funksioni i kontrollit të gjendjes do të kthejë apo cojë në gjendjen FACH, përndryshe në atë DCH.

6.3.4 Problematika të cilësisë që krijohen zgjidhja

Meqënëse qëllimi i modulit tonë është planifikimi i transmetimit të të dhënave nëpërmjet zgjedhjes së transferimit në kanale trafiku dhe vonimit të trafikut në rastet e komunikimeve (sepse paketat vendosen në pritje dhe transmetohen pas një kohe pritjeje), do të sjell efekte në mënyrën e operimit të disa aplikimeve dhe protokolleve të rrjetit të cilët janë të ndjeshëm ndaj vonesave. Moduli i ri apo patchi sw është krijuar me qëllimin për aplikimet që veprojnë në sfond / background por realisht ai zgjedh të gjithë trafikun dalës apo atë në drejtimin UL pa ndonjë dallim. Por, megjithatë të gjitha aplikimet nuk ndajnë llogjikën e tyre të komunikimit nëse kanë ose jo trafik background, dhe varësia me të lind nga tipi i aplikimeve dhe protokollet e rrjetit që përdorin. Gjatë matjeve për filtrimin e paketave gjatë komunikimeve përdoret aplikimi e kapjes së paketave (tshark apo bitshark, wireshark etj) ku secili filtron paketat e marra & dërguara nga dhe drejt portave specifike. Por mund të përdoren dhe fila logger në smartphone. Për secilën prej këtyre aplikimeve telefonat Android duhet të jenë root-ed (pra me të drejtën e egzekutimit të komandave si su – superuser, por që në Android Linux-i i tij nuk e gëzon këtë të drejtë dhe prandaj përdoren aplikime si SuperSU dhe CWM). Aplikimet Voip si dhe ato të rrjetave sociale si Facebook përdorin protokollin TCP. TCP i cili është një protokoll i cili siguron që të dhënat të shpërndahen në destinacion pa gabime dhe në të njëjtën mënyrë sikur u transmetuan. Lidhjet TCP kanë tre faza: *stabilizim, transmetim të dhënash* dhe *përfundimi.* Në fazën e transmetimit, mekanizma të ndryshëm janë përdorur për të kontrolluar rrjedhjen e paketës, si numri i sekuencës, checksum, njohja e kohëzuesve etj [95]. *Nr i sekuences*: ai lejon marrjen sipas renditjes të paketave TCP dhe dedekton paketat e dublikuara. *Checksum*: ai dedekton segmentet për ndonjë gabim. *ACK*: ai kërkon nëse ndonjë paketë ka humbur. *Kohëzuesit*: vendos nëse një paketë duhet të ritransmetohet apo jo. Bazuar në këto mekanizma, paketat që nuk kanë ndjekur komunikimin TCP, dedektohen dhe futen në një nga klasifikimet si: Ritransmetim TCP, TCP jashtë rradhe dhe TCP dublikim ACK.



- | |

Figura 6.6: Shkëmbimi i mesazheve TCP në komunikim midis 2 nyjeve [93]

Në rastin e parë, transmetuesi aktivizon një kohëzues RTO çdo herë që një paketë dërgohet dhe nëse RTO ka kaluar afati kohor (llogaritur si rrjedhojë e RTT që përfytyrohet si kohë që nga dërgimi i paketës deri në marrjen e ACK) paketa do të ritransmetohet. *Paketa TCP jashtë rradhe*: Një paketë TCP është jashtë-liste kur Byte-et nuk korrespondojnë me numrin tjetër të njohjes. Kjo nënkupton që paketa është shpërndarë/ marrë në renditje/rradhë tjetër nga sa ajo është dërguar. *Paketë TCP e dublikuar ACK*: Një paketë e duplikuar TCP ACK është dërguar kur një paketë jashtë rradhe është marrë. Figura 6.7 tregon një shembull për shumën e këtyre paketave TCP brenda trafikut real (p.sh një trace nga WIreshark).

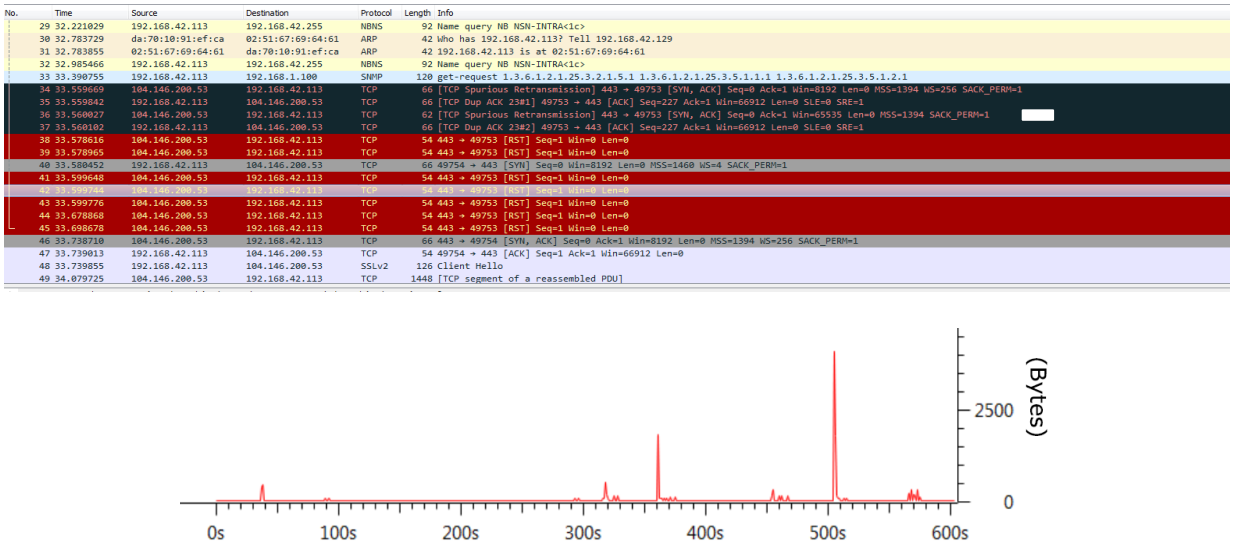


Figura 6.7: Shembull paketash të ritransmetuara, ACK dublikuara dhe jashtë rradhe

Kjo mund të perceptohet dhe si ndërprerje apo shkëputje e lidhjes. Për këtë gjë, implementohet llogjika e "TCP Keepalive" që nënkupton kontrollimin nëse një lidhje TCP (TCP socket) është e ngritur apo është rrëzuar. Megjithatë egzistojnë dhe një sërë parametrash që mund të përmirësojnë disi këto problematika (në shtresën 2-3 të TCP/IP) [30, 93, 95].

6.3.4.1 Mesazhet TCP keep alive

Koncepti i "KeepAlive" është shumë i thjeshtë: kur krijojmë një lidhje TCP, ju i shoqëroni një sërë kohëzuesish lidhjes. Disa nga këto kohëzues kanë të bëjnë me procedurën e "KeepAlive". Kur kohëzuesi keepalive arrin në vlerën zero, i dërgohet nyjës të anës tjetër të komunikimit një paketë keepalive bosh dhe flamuri ACK ON në të. Kjo mund të bëhet për shkak të specifikave të TCP / IP, si rrjedhojë e dublikimeve ACK, dhe "endpoint" / nyje fundore i largët nuk do të ketë argumente, sepse TCP është një protokoll i orientuar stream. Nga ana tjetër, do të marrim një përgjigje nga nyja në distancë (e cila nuk ka nevojë për të mbështetur mesazhet *KeepAlive* në përgjithësi, por vetëm TCP/IP), me një paketë bosh / pa të dhëna dhe ku ka vendosur ACK.

Nëse do të marrim një përgjigje për "probe-in" tonë *KeepAlive*, ne mund të pohojmë se lidhja është ende e ngritur dhe nuk shqetësohemi për zbatimin në nivelin e përdoruesit. Në fakt, TCP na lejon

të trajtojmë në nivel stream, dhe jo paketë dhe kështu që një paketë me madhësi "zero" nuk është e rrezikshme për aplikimin e përdoruesit. Kjo proçedurë është e dobishme, sepse nëqoftëse nyjet e tjerë humbin lidhjen e tyre (për shembull nga një "rebooting") do të vemë re se lidhja është shkëputur edhe nëqoftëse nuk kemi trafik të dhënash në të. Nëse "probe-t" KeepAlive nuk i janë përgjigjur nyjet fqinje, ne (nyja) mund të pohojmë se lidhja nuk mund të konsiderohet e vlefshme dhe pastaj të marrim veprimin e duhur. Ne do të bëjmë dallimin në mes këtyre dy detyrave të synuara për "keepalive":

- Kërkimi për nyje të "vdekura" / shkëputura.
- Parandalim i shkëputjes si pasojë e pasivitetit

Në interesin tonë është për të dytën pasi qëllimi i planifikimit të transmetimeve dhe vonesat në transmetim mund të çojnë dhe në "shkëputje" si shkak i vonesave në komunikim dhe i mospërgjigjeve në mesazhet handshake (ACK).

6.3.4.2 Parandalimi i shkëputjeve si shkak i pasivitetit të rrjetit

Qëllimi tjetër i dobishëm i keepalive është të parandalojë pasivitetin nga shkyçja e kanalit. Kjo është një çështje shumë e zakonshme, kur jemi pas një proxy NAT ose një firewall, dhe të mund të jemi të shkëputur pa arsye. Kjo sjellje është shkaktuar nga proçedurat e ndjekjes/tracking së lidhjes të zbatuara në Proxy-t dhe firewall-et, të cilat mbajnë gjurmë të të gjitha lidhjeve që kalojnë përmes tyre. Për shkak të kufizimeve fizike të këtyre "makinave", ata mund të mbajnë vetëm një numër të caktuar të lidhjeve në kujtesën e tyre. Mënyra më e zakonshme dhe e logjikshme është që të mbajë lidhje të reja dhe të hidhen lidhjet e vjetra dhe joaktive të mëparshme.

Duke e rikthyer në nyjet e A dhe B, rilidhjen e tyre: pasi kanali është i hapur, presim derisa ndodh një ngjarje dhe pastaj ia komunikojmë këtë nyjes tjetër. Çfarë ndodh nëse ngjarja verifikohet pas një periudhe të gjatë kohore ?. Lidhja jonë ka qëllimin e saj, por është e panjohur për Proxy-in. Pra, kur ne më në fund dërgojmë të dhëna, proxy nuk është në gjendje ta trajtojë siç duhet atë dhe lidhja prishet ose shkëputet. Egzistojnë një sërë parametrash për të cilat mund të sjellin një optimizim të protokollit TCP/IP në varësi të kushteve:

Pavarësisht se zgjidhja që ne duam të arrijmë ka një performancë të mirë për protokollin UDP, ajo nuk mund te japë të njëjtën gjë për atë TCP, ku sigurisht mund të krijojë problematika të ritransmetimeve etj.

TCP_USER_TIMEOUT (që prej Linux 2.6.37)

Ky opsion merr një vlerë *unsigned_int* si një argument. Kur vlera është më e madhe se 0, kjo përcakton sasinë maksimale të kohës në milisekonda që të dhënat e transmetuara mund të mbeten "të papranuara" / panjohura përpara se TCP të mbyllë me forcë lidhjen përkatëse dhe të kthejë një **ETIMEDOUT** te aplikacioni. Nëse opsioni i vlerës është specifikuar si 0, TCP do të përdorë vlerën default të sistemit. Rritja e "timeouts" për përdoruesit lejon që një lidhje TCP të mbijetojë për periudhë më të zgjeruar dhe pa lidhjen fund-më-fund. Duke zvogëluar kohën "timeouts" të përdoruesit, kjo lejon që aplikimet të "dështojnë më shpejt", ose në të kundërt të pritet ende. *Përndryshe, dështimi mund të zgjasë deri në <u>20 minuta</u> me vlerat aktuale default të sistemit në një mjedis normal WAN.*

Ky opsion mund të jetë vendosur gjatë çdo gjendjeje të një lidhje TCP, por është efektive vetëm gjatë gjendjeve të sinkronizuara të një lidhje (ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, and LAST-ACK). Për më tepër, kur përdoret me opsionin TCP keepAlive (**SO_KEEPALIVE**), **TCP_USER_TIMEOUT** do anashkalojë KeepAlive për të përcaktuar se kur do të mbyllë një lidhje të duhur për shkak të një dështimi KeepAlive. Opsioni nuk ka efekt kur një TCP ritransmeton një paketë, dhe kur një paketë testi / probe është dërguar. Ky opsion si shumë të tjerë, do të trashëgohet nga një socket i kthyer nga mszh accept (2), nëse ajo është vendosur si një socket dëgjuese. Detaje të mëtejshme në këtë shërbim të ri të "timeout" të përdoruesit mund të gjendet në specifikimet RFC 793 dhe RFC 5482 ("TCP User Timeout Option") [93].

TCP_NODELAY
Nëse është vendosur, çaktivizon algoritmin Nagle. Kjo do të thotë se segmentet janë dërguar gjithmonë, sa më shpejt që të jetë e mundur, edhe nëqoftëse ka vetëm një sasi të vogël të të dhënave. Kur nuk janë caktuar, të dhënat janë "bufferuar" deri sa të ketë një sasi të mjaftueshme për të dërguar jashtë, duke shmangur dërgimet e shpeshta të paketave të vogla, e cila rezulton në një shfrytëzim "të varfër" të rrjetit. Ky opsion, anashkalohet nga **TCP_CORK**. Megjithatë, duke vendosur këtë opsion i detyron transferimi e një fluksi të lartë të daljeve (Tx) të vëna në pritje, edhe nëqoftëse **TCP_CORK** është vendosur aktualisht.

tcp_frto (integer)
Mundëson një Forward rimëkëmbje-RTO (F-RTO) përcaktuar në specifikimin RFC5682. F-RTO është një algoritëm i zgjeruar i rimëkëmbjes për ritransmetimin e TCP timeout / kohëve limit. Kjo është veçanërisht e dobishme në rrjetet ku RTT luhatet (psh, wireless). *F-RTO është vetëm i modifikuar në anën e dërguesit*. Ajo nuk kërkon ndonjë mbështetje nga ana tjetër marrëse / peer. Si default është e lejuar me një vlerë jo-zero dhe për më shumë vlera: 0-bllokon F-RTO. 2 – mundëson një mszh SACK të zgjeruar të F-RTO nëse fluksi përdor mesazhet SACK.

tcp_frto_response (integer; vlera default: 0, që prej Linux 2.6.22).
Kur F-RTO ka dedektuar që koha limit e ritransmetimeve TCP është false, TCP ka një sërë opsionesh për të vepruar më pas. Vlera 0 – për një përgjigje të butë e konservative. 2- përgjigje më aggresive (nuk rekomandohet).

tcp_low_latency (Boolean; vlera default "dissabled", që prej Linux 2.4.21 / 2.6) Nëse aktivizohet, staku TCP merr vendime që preferojnë vonesa më të ulëta në krahasim me një ngarkesë të lartë. Nëse ky opsion është caktivizuar, atëherë një ngarkesë më e lartë është e preferuar [30].

tcp_min_rtt_wlen (integer, default: 300)
Gjatësia e dritares së filtrit min për të gjetur RTT minimale. Një dritare e shkurtër lejon një fluks më shpejt për të marrë një RTT minimale të re (më i lartë) kur ajo është zhvendosur në një rrugë më të gjatë (p.sh., për shkak të trafikut inxhinierik). Një dritare më e gjatë bën filtrin më rezistent ndaj luhatjeve të RTT si kongjestione kalimtare. Njësia është sekonda.

Ajo që në dallim me punimet e mëparshme është që të vendosim që këto parametra të aplikohen sikur ne duam në boot apo kenrelin (SW të UE me bazë linux - Android) duke i shtuar ato te fila në: `/etc/sysctl.conf`:

/etc/sysctl.conf

```
1 # parametra TCP stack  per nje rrjet Wireless
2 net.ipv4.tcp_frto = 1
3 net.ipv4.tcp_frto_response = 0
4 net.ipv4.tcp_low_latency = 0                  // e caktivizuar
```

6.3.4.3 Ndërveprimi me sistemet reale (aplikimet foreground)

Aplikacionet që i presin përgjigjet në kohë reale mund të reagojnë keq në lidhje me algoritmin Nagle. Aplikime të tilla si video games në rrjet presin që veprimet në lojë të dërgohen menjëherë, ndërsa algoritmi qëllimisht vonon transmetimin, rrit efikasitetin e bandwidth-it në kurriz të vonesave. Për këtë arsye, aplikimet me bandwidth të ulët e kohë të ndjeshme të transmetimeve zakonisht përdorin TCP_NODELAY për të anashkaluar vonesën nga algoritmi Nagle. Në fund të fundit mekanizmat e TCP dhe IP (shtresat Transport dhe Rrjet) janë për të siguruar ofrimin e trafikut në një drejtim (nga burimi në destinacion) me disa kufizime në fushën e ri-tentimeve maksimale etj. Komunikimi i aplikimeve është në fund të fundit një lidhje e dyfishtë e plotë për komunikimet në shtresën e Aplikimit që qëndron në krye të TCP / IP. Ngatërrimi i këtyre shtresave me njëra-tjetrën apo funksionet e tyre e thënë ndryshe, nuk është një strategji e mirë.

Por edhe për shkak se TCP ka për qëllim që stream-et të kalojnë nga një burim te një destinacion, dhe jo të sigurojë në të vërtetë një komunikim të plotë-të dyfishtë. Njohjet e ACK-t të nivelit TCP jo domosdoshmërisht i korrespondojnë llogjikës 1-1 për secilin nga transmetimet e aplikimeve të rrjetit (dhe pa dyshim nuk do të jetë 1-1 në qoftë se dërgojmë më shumë se madhësinë maksimale të MTU apo paketave të rrjetit, edhe në qoftë se Nagle është bërë “disabled” (i çaktivizuar).

Pajisjet apo UE shpesh marrin më mirë se ata që mund të transmetojnë. Është e zakonshme për pajisjen që të marrë një dërgesë të përkryer dhe të përgjigjet me një lloj "ACK", e cila është transmetuar, por mund te ndodh që një ACK wireless nuk arrin destinacionin e vet për shkak të cilësisë së sinjalit, distancës së transmetimit, interferencave RF, humbjeve të sinjalit, reflektimeve të sinjalit, etj. Këtu duam të bëjmë një ngjashmëri me idenë tonë me rastet kur pajisja apo UE i vë paketat në pritje dhe i vonon ato (qofshin dhe ACK nga UE). Në çfarë pike në këtë skenar klienti merr një veprim (të ruaj të dhëna apo të dërgojë ato duke ndryshuar kanal) dhe në çfarë pike serveri konsideron që aksioni është i suksesshëm ?. Kjo është e pamundur të zgjidhet në mënyrë të besueshme pa një komunikim shtesë dhe ndërveprime apo kërkime nga shtresa e aplikimit (duke përfshirë mënyrën dy-drejtimëshe të njohjeve të mesazheve ACK).

Në përgjithësi nuk duhet ti besojmë njohjeve ACK në shtresën TCP sepse nuk i përgjigjet në mënyre të barabartë një komunikimi dupleks të plotë (sikur bëhet në shtresën e aplikimit) dhe nuk do të lehtesojë një mekanizëm të besueshëm të tentativave (për përseritje) të aplikimeve.

Një tjetër alternativë është që të përdoret UDP në vend të TCP. Por kjo mbetet në dorë të programuesve të aplikimeve.

6.4 Mjedisi vlerësues për zgjedhjen e transferimeve

Mjedisi vlerësues përbëhet nga setup-i i matjeve fizike dhe një server UDP në internet (apo një PC ose UE me një aplikim specifik UDP për dërgim e marrje paketash). Serveri apo aplikimi UDP është përdorur për të gjurmuar dhe siguruar që të gjitha paketat e destinuara janë dërguar me të vërtetë nga setup-i ynë i matjeve fizike.

Tracet janë fila *.pcap* (paketë e kapur) që përbëhet nga trafiku i paketave të të dhënave.Wireshark ose aplikime në UE Android përdoren për mbledhjen dhe analizimin e filave të matjeve në format *.pcap* në sekuenca të paketave të dërguara me formatin e mëposhtëm: *ftimestamp*, *packetsize*, *packetnumber*, *burim*, *destinacion*, *protocolg*. Në bazë të këtyre të dhënave mund të filtrohet dërguesi nga marrësi, drejtimi i paketave në Tx apo Rx dhe kohët për secilën, si dhe fazat e komunikimit.

Dërgimi i paketave me madhësi të ndryshueshme mund të bëhet dhe nga një PC me *cmd.exe* ose direkt nga telefoni në Android duke përdorur një api "terminal" duke përcaktuar:

```
C:> ping xxx.xxx.xxx.xxx –l 200 –n 10
C:> ping xxx.xxx.xxx.xxx –l 250 –n 10
C:> ping xxx.xxx.xxx.xxx –l 300 –n 10

user@android:/$ ping xxx.xxx.xxx.xxx –l 200 –n 10
user@android:/$ ping xxx.xxx.xxx.xxx –l 250 –n 10
user@android:/$ ping xxx.xxx.xxx.xxx –l 300 –n 10
```

E nëse tranzitimi vërehet midis 250 – 300 Byte atëherë tentojmë me numër me të vogël paketash duke nisur po nga 250 +10, e kështu me radhe derisa të gjejmë vlerën e duhur të tranzitimeve.

```
C:> ping xxx.xxx.xxx.xxx –l 260 –n 10

user@android:/$ ping xxx.xxx.xxx.xxx –l 260 –n 10
```

apo UEAndroid root-ed:

```
user@android:/# ping xxx.xxx.xxx.xxx –l 260 –n 10
```

Parelelisht me to mund të vërehen dhe kohëzuesit e mosaktivitetit. Një mënyrë tjetër është dhe përdorimi i Nokia E6 me aplikimin NEP apo dhe api powertutor ne Android i cili shfaq grafikisht konsumin e fuqisë në telefon. Duhet përjashtuar nga ato vlera fuqia e kërkuar për ekranin dhe aplikimin NEP (diskutuar në kapitullin 5, figura 5.14). Ndërkohë një metodë shumë eficente është dhe përdorimi në PC i aplikimit LittleEye Labs i cili është dizenjuar dhe për të matur energjinë e konsumuar në një smartphone me bazë Android të lidhur me USB me PC [144]. Gjithashtu ai ndihmon në matjen e konsumit për aplikim, CPUs, ndërfaqe etj. Ky aplikim kërkon që të keni të instaluar Android Studio e SDK si dhe filën *guava-xx.jar* të instaluar në /sdk/tools/lib/. Aktualisht ky është një tools nën menaxhimin e Facebook tashmë dhe nuk është më në përdorim për përdorues apo testues të tjerë në këtë fushë.

6.4.1 Ndikimi i modulit në konsumin shtesë të CPUs

Një faktor i rëndësishëm është dhe vlerësimi i shtesave në punën apo funksionin e CPUs pasi në këtë mënyrë mund të dihet apo nxirret në pah energjia shtesë e kërkuar nga moduli i ri për funksionimin e tij. Të gjitha aplikimet që janë në telefon janë bllokuar për të qenë të ekzekutueshëm në telefon dhe matjet në telefon janë kryer duke pasur në konsideratë faktin që ekrani dhe të gjitha ndërfaqet e rrjetit janë bllokuar / fikur në mënyrë që të izolohet çdo ndërveprim i mundshëm i tyre në punën shtesë të CPU. Për më shumë shpejtësia e CPUs mund të vendoset në maksimum me një aplikim (AnTuTu CPU master) në mënyrë që të kryhen matjet për rastin e konsumit me energji më të lartë. Aplikimi mund të gjenerojë një ngarkesë të CPU's me përqindje të ndryshme nga 5 deri në 100% (me nga 15 deri 25% ndryshim ngarkese). Vendosim CPU në punë ose jo me një ngarkesë (me anë të aplikimit), psh. në 300ms të kohës të jetë me ngarkesë dhe për 700ms të jetë pushim. Për të arritur ngarkesa të ndryshme, duhet të ndryshojmë periudhën e kohës gjatë së cilës CPU punon. Figura 6.8 tregon një mesatarizim të konsumit të fuqisë së CPU-s.

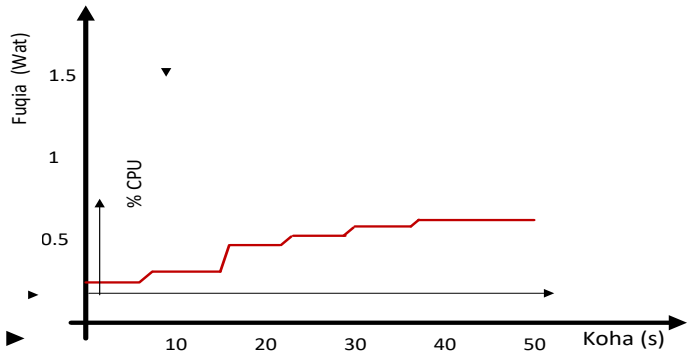


Figura 6.8: Konsumi mesatar i fuqisë së CPU-s me aplikimin testues

Ngarkesa e CPU	Fuqia konsumuar (Wat)
5%	0.26
15%	0.32
30%	0.5
50%	0.52
75%	0.61
100%	0.72

Tabela 6.1: Konsumi i fuqisë së çastit për ngarkesë të CPU

Testimet për nxjerrjen e konsumit ekstra të CPU-s, kur moduli proceson funksione të menaxhimit të rradhëve, të bëhen duke lidhur telefonin nën testim në një laptop nëpërmjet një fishe USB dhe duke mos lejuar që telefoni / UE të marrë energji nga Laptopi. Telefoni sikur përmendëm do të jetë me ekranin të fikur dhe pa aplikime në të (të bllokuara). Qëllimi është të ekzekutojmë disa paketa UDP të cilat procesohen nga moduli duke u vënë në pritje edhe më pas ato transmetohen. Matjet bëhen duke përdorur një tools matjeje direkt në baterinë e telefonit apo një prej tools-eve në tabelë.

Pjesa e procesimit të pritjes së paketave është dhe pjesa ku moduli merr pjesë në ngarkesën shtesë të CPUs dhe nga matjet vërehet një shtesë e papërfillshme e shtesës së fuqisë prej disa dhjetra mW dhe në pjesë të caktuara të kohës kur nga moduli kryhen procese kontrolli apo llogaritjeje:

Fuqia aktuale konsumuar (mW)	**Shtesa e Fuqisë në CPU (impulse)**
> 600 mW	~ 30 mW

Tabela 6.2: Konsumi i fuqisë shtesë të CPU

Në grupin e testimeve mund të testojmë me trafik të simuluar (jo real apo vetëm për testime) dhe ku volumi i trafikut me dhe pa modulin është i njëjtë, si dhe një trafik real apo me aplikim në sfond të cilat ekzekutohen pa dëshirën e përdoruesit.

6.4.2 Matjet nga trafiku real i stimuluar

Në fazën e mbledhjes së të dhënave, kemi përdorur *tcpdump* në një smartphone Samsung Galaxy S3/4 me sistem operativ Android me linux kernel 2.3.7 lidhur me të njëjtin operator mobile 3G e 4G në Shqipëri dhe Lab jashte Shqipërie duke përdorur karta USIM. Në mënyrë që të lejojmë vetëm trafikun nga aplikacionet e dëshiruara kemi përdorur një *iptables firewall* bazuar në aplikimin DroidWall. Ne kemi kapur të gjithë komunikimin e 3G/4G për disa minuta (10 – 30 minuta). Aplikacione të ekzekutuar në smartphone është ai per testime *UDP, GooglePlay, Gmail, Facebook e AccuWeather*. Të gjitha tracet janë mbledhur me ekran fikur dhe aplikimet janë aktive në sfond/background. Nuk kemi asnjë ndërveprim përdoruesi gjatë matjeve. Tracet e mbledhur kanë shumë paketa të vogla.

Smartphon-i nën testim ekzekuton një aplikim klienti UDP, që dërgon trafik real të emuluar te serveri duke përdorur ndërfaqen 3G në UE. Firewalli (DroidWall) është përdorur për të lejuar vetëm aplikimin nën testim për të dërguar të dhëna nëpërmjet 3G/4G dhe të shmangi gjithë trafikun tjetër. Serveri / UE në anën tjetër në mënyrë simultane ekzekuton aplikimin UDP Server, që i përgjigjet me një “echo UDP” çdo pakete të marrë nga klienti (UE në rastin tonë). Matja është kryer me dhe pa modulin e ri në OS të UE. Figura 6.9 përshkruan setup-in për kryerjen e matjeve.

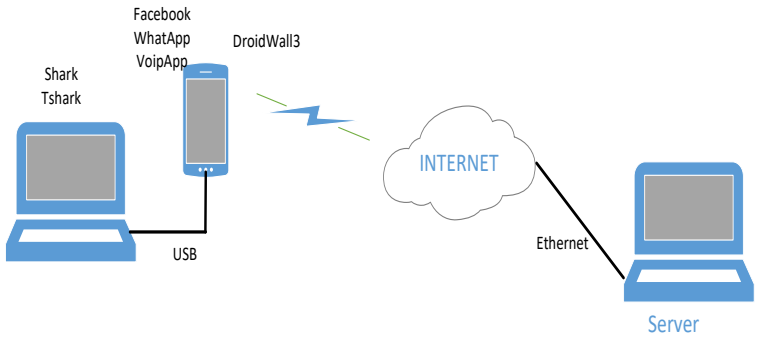


Figura 6.9: Setup-i per të matur ruajtjen e energjisë së trafikut real të stimuluar

Ky trafik real i stimuluar, simulon trafikun real të gjeneruar nga dërgimi i paketave UDP. Në mënyrë që të gjenerojmë trafik real të matshëm, në i ekzekutojmë këtë aplikim UDP në telefon dhe PC / UE në anën për të gjeneruar trafik real dhe përdorim aplikimin për kapje paketash në PC

ose në UE (tshark apo tpacketcapture) për të kapur trafikun gjatë një kohë të caktuar. Aplikimi mbledh një file *.pcap* nga tracet e mbledhura që specifikon madhësinë e çdo paketë, nëse paketa i përket Uplinkut/UL apo Downlinkut/DL dhe kohën në të cilën është transmetuar / marrë. Ne ekstraktojmë trafikun UL të këtij file .pcap, dhe e lexojmë atë me një aplikim Wireshark në PC apo qoftë dhe UE. Në këtë mënyrë kemi matur një trafik "real".

Ne kemi gjeneruar trafik të ndryshëm, të matshëm e real. Tracet janë gjeneruar me përdorues aktive në ekran duke ndërvepruar mbi aplikimet si dhe në rastin e dytë janë gjeneruar pa një ndërveprim të përdoruesit dhe kur ekrani është i fikur i lidhur në PC. Në këtë mënyrë, ne shmangim trafikun e zërit/thirrjet dhe SMS-të, si dhe simulojmë trafikun sfond / background.

6.4.3 Gjenerimi i të dhënave matese nga trafiku real

Për të relizuar këtë gjë ne duhet të gjenerojmë trafik real ku shërbimet e aplikimeve janë të ekzekutueshme në telefon dhe ku ne nuk e kontrollojmë trafikun dy-drejtimësh apo dhe funksionin e aplikimeve nën testim. Për këtë gjë ne mund të zgjedhim të testojmë një aplikim si Facebook dhe për këtë gjë përdorim aplikimin DroidWall për të lejuar vetëm trafikun e këtij aplikimi. Realizimi i matjeve kryhet në mënyrë të thjeshtë me pajisjet: një smartphone me aplikimet e nevojshme të lidhur me USB me një PC si në figurën e mëposhtme:

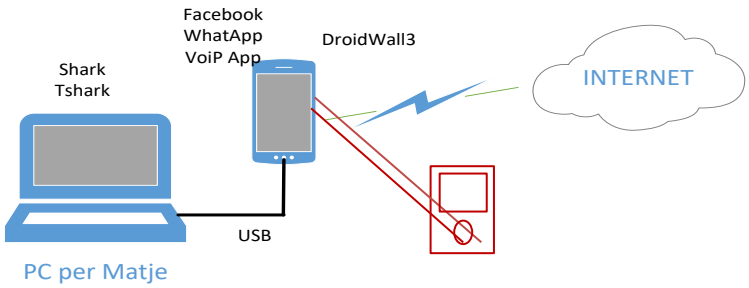


Figura 6.10: Testimi me anë të trafikut real

Testimet janë kryer për aplikime të ndara, në veçanti për një periudhë kohore prej disa sekondash deri disa minutash dhe me aplikimin e lejuar në background, pa dhe me modulin si dhe të ekzekutueshme në sfond vetëm për të përjashtuar ndërveprimin e përdoruesit. Për më tepër, në lidhje me rezultatet e matjeve do të flitet më tej në kapitullin 7 në seksionin e matjeve dhe rezultateve.

Mënyra e realizimit të testeve sipas figurës 6.9 duhet të jetë në sinkron me kushtet e kërkuara në modulin e planifikimit të transmetimeve. Aplikime si Wireshark në PC e UE si dhe aplikime ndihmëse apo module HW të lidhur me baterinë e UE si dhe Vm për matje rënie tensioni konsiderohen gjatë proçesit të matjeve.

KAPITULLI VII

7 ANDROID DHE IMPLEMENTIMI I ZGJIDHJES

7.1 Sistemi operativ mobile Android

Ne do të fokusohemi te rasti i sistemeve operative mobile, si Android OS (2008). Android-i është një nga sistemet operuese më popullore për pajisjet moderne smartphone dhe tabletet. Ai është krijuar nga kompania Google disa vjet më parë, dhe ekziston në më shumë se gjysmën e pajisjeve "të mençura" (smart) sot në Amerikë dhe së shpejti në gjithë botën. Eshtë një sistem i hapur në burim kështu që është përdorur nga shumë zhvillues dhe prodhues UEs, për të modifikuar funksionalitetet dhe/ose shtuar të reja.

Ka disa arsye pse është zgjedhur ky sistem dhe pikësëpari është shpërndarja dhe përdorshmëria shumë e lartë e këtij OS në smartphonet e sotëm [ABI Research, 2014 - 2016]. Së dyti, Android është një sistem OS i hapur por jo dhe aq sa mund të mendohet pasi dhe ai ka limitet e tij. Psh SW't e driverave janë "të mbyllur" dhe kjo në varësi të prodhuesve duke bërë të mundur mbrojtjen e të ardhurave/fitimeve në këtë pjesë. Por shumica e kodit burim të Android është i liçensuar nën licensën Apache. Kodi i tij është i mundur të merret "online" dhe i mjaftueshëm për studim për një grup më të gjerë. Megjithatë dizenjimi i Android tregon se sistemi Android nuk i beson plotësisht kernelit të tij. Për të zgjeruar pavarësinë, *Androidi i ekzekuton të gjitha aplikimet e tij në një makinë virtuale Dalvavik dhe detyron izolimin midis aplikimeve dhe shërbimeve lokale të sistemit.* Një proces special, Zygote, është krijuar gjatë inicializimit për të monitoruar kontrollin e aksesit të çdo komponenti të sistemit. Këto zgjidhje mbështeten në integritetin e kernelit të Android.

Platforma Android mund të perceptohet si një software stack që përfshin një sistem operimi, pjesën middleware dhe aplikimet. Sistemi operativ është i bazuar në një Kernel të Linuxit, i cili është përgjegjës për abstraksionet HW të tilla si driverat, menaxhimin e kujtesës, menaxhimin e proceseve, të rrjetit dhe disa kontrolle të tjera të sigurisë (si menaxhimin e përdoruesit dhe aksesimin e filave). Ai përdor strukturën kernel të Linux-it me modifikime si *wakelocks, binder IPC, lowmem killer* dhe një numër të madh ndryshimesh që e përshtasin në një pajisje moderne celulare.

Shtresa *middleware* është përgjegjëse për të siguruar shërbimet e kërkuara nga shtresa e aplikimit. Ajo përmban bibliotekat amtare Android, që zbatojnë disa funksionalitete të ndryshme të tilla si aksesi i bazës së të dhënave, shfletues të integruar, mbështetjen media, akses në ndërfaqen grafike dhe të tjera. Në krye të saj është shtresa aplikative për aplikimet e krijuar / hartuar / kompiluar nga kodet Java. Kjo shtresë mund të ndahet në frameworkun e aplikacioneve dhe aplikimin. Figura 7.1 tregon arkitekturën aktuale të stakut OS të Android.

Frameworku i Android konsiston në libraritë që kanë një sërë funksionalitetesh të ndryshme. Në qendër të saj është makina virtuale Dalvavik që është esenciale për gjithë aplikimet. Aplikimet Android shkruhen në Java të cilat më pas "përpilohen" në *bytecoded dalvavik* për interpretim nga VM (makina virtuale). Frameworku Android ofron shumë funksionalitete përmes biblotekave të

ndryshme Java që qëndrojnë midis aplikimit dhe VM. Këto janë analoge me bibliotekat që janë *bundled/te grupuara* së bashku me Java JDK. Këto biblioteka janë burim i hapur dhe i lejojnë përdoruesve të inspektojnë dhe modifikojnë sjelljen e tyre. Siç u përmend më herët, një aplikim shkon në krye të Dalvik VM, dhe përdor bibliotekat e ndryshme siç është paraqitur në figurën 7.1. Sistemi i biblotekave Server ndërvepron me ndërfaqet në dispozicion të rrjetit në një pajisje të lëvizshme, menaxhon ato dhe jep informacion në lidhje me to te "Apps". Biblioteka *libcore* siguron funksionalite të lidhjes së rrjetit në shtresa të ndryshme të tilla si Socket, HTTP e të tjera.

Mënyra standarde për zhvillimin e aplikacioneve për platformën Android është nëpërmjet përdorimit Java dhe XML, por ka edhe një mjedis/tool për zhvillimin e Android duke përdorur gjuhë programimi amtare si C dhe C ++ të quajtur Kit Development Native (NDK) [90]. Një program i hartuar Java mund të ekzekutohet në çdo platformë që ka një JVM, por për fat të keq Java ka një reputacion performancë të dobët. *Kodet amtare në C / C ++ megjithatë nuk kanë nevojë për një makinë virtuale, përderisa ato janë hartuar (compiled) në kodin e makinës për të vepruar/ekzekutuar në një procesor të veçantë dhe është në teori më e shpejt.*

7.1.1 Arkitektura e platformës Android

Struktura software e Android përbëhet nga katër përbërës kryesore: Sistemi Operativ (OS), Middleware, Korniza e aplikimit dhe Aplikimet, e për më shumë shih Figurën 7.1.

-***Sistemi operativ:*** Android OS mbështetet në një modifikim të Linux Kernel 2.6.x apo dhe ato si 3.0.x projektuar që të optimizojë memorien dhe burimet hardware në mjedis të lëvizshëm. Kjo shtresë vepron si një shtresë abstraksioni në mes të hardware dhe pjesës tjetër të stak-ut software /strukturës.

-***Middleware:*** kjo shtresë përbëhet nga Runtime Android (RT) dhe një sërë bibliotekash të shkruara në C / C ++ që ofrojnë shumicën e tipareve të sistemit. Android RT është formuar nga makineri virtuale Dalvik (VM) dhe një sërë funksionesh amtare/bazë të Android. Dalvik VM është një makinë virtuale me regjistër bazë dhe optimizuar për pajisje të lëvizshme.

-***Kuadri i aplikimit:*** kjo kornizë siguron një grupim të mjeteve të zhvillimit / toolse që përdoren nga aplikimet, nëse ata janë pjesë e pajisjes amtare/bazë ose të zhvilluara nga një palë e tretë.

-***Aplikimi:*** kjo është shtresa ku vendoson aplikacionet. Aplikacionet janë të shkruara në gjuhën e programimit Java dhe ata përdorin API-t dhe bibliotekat e shtresave më të ulëta.

Izolimi i Aplikimeve: Çdo hapësirë aplikimi është izoluar nëpërmjet një makine virtuale Dalvik. Kjo do të thotë se një aplikim nuk mund të hyjë / aksesojë direkt në një aplikim tjetër ose hapësirat e OS. Në qoftë se një aplikim është *fals* ose crashes (dështon), dështimi nuk do të propagandohet në aplikimet e tjera.

Kerneli Linux: Android është ndërtuar në majë të një kerneli Linux. Çdo aplikim është trajtuar si një përdorues tjetër (UID) dhe për disa burimeve (p.sh. internetit, bluetooth) janë aplikuar lejimet (permission) skematike të Linux. Edhe pse ajo është ndërtuar në majë të kernel Linux, Android

nuk është identike me Linux, pasi ai nuk përfshin seri të plotë të shërbimeve standarde Linux. kerneli është përgjegjës për bashkëveprime me hardware të pajisjes dhe përmban të gjitha driverat thelbësore hardware / fizik.

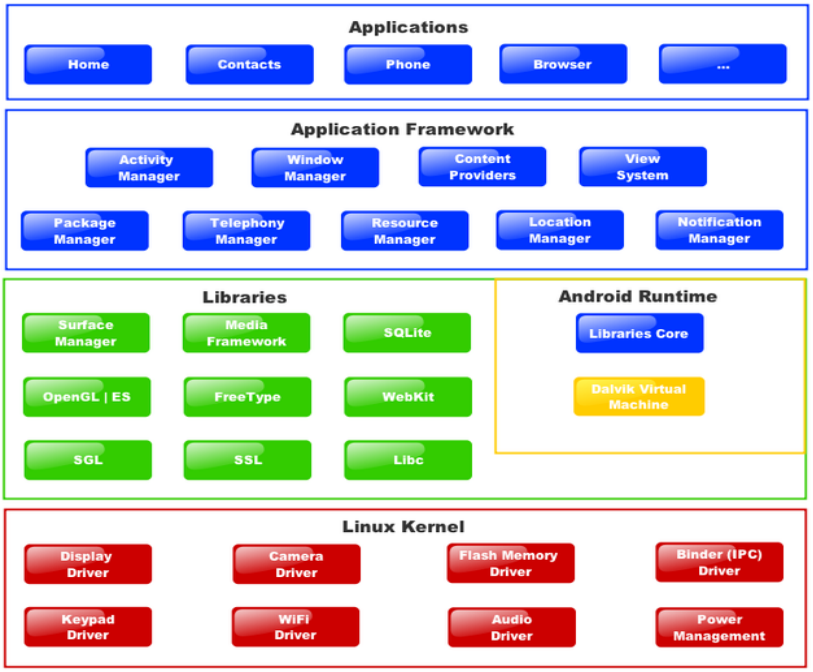


Figura 7.1: Staku i sistemit operativ mobile Android [90]

Tools-i Android SDK është përgjegjës për hartimin e kodit në një paketë Android, i cili është një file me prapashtese.*apk*. Kodi si një file *.apk* është konsideruar të jetë një aplikim i vetëm për pajisjet bazë Android që përdorin këtë file për të instaluar aplikacionin aktual në kujtesën e brendshme të pajisjes.

Në frameworkun e aplikimit gjenden disa nga tools-et e Android dhe shërbimeve të tilla si ofruesit e përmbajtjes, menaxheri i aktivitetit, menaxheri i telefonisë, menaxheri i vendndodhjes dhe të tjera. Në krye të tyre është shtresa aplikative ku janë aplikimet e përdoruesit si telefoni, e-mail klient, shfletuesi i web, lojra dhe të tjera. Të gjitha aplikacionet janë të izoluar në hapësirën e tyre, ekzekutojnë proceset në vetvete/sipas tyre, unik, dhe privilegje të niveleve të ulëta të përdoruesve id, në krye të shkallës / instancës së tyre të një makine Dalvik Virtual. *Në qoftë se një aplikim është fals ose bën crashes/bie, dështimi nuk duhet përhapur te aplikimet e tjera.* Ka leje/permission të veçanta për secilin burim që është e nevojshme për një aplikim. Kërkesat për leje që nuk janë dhënë në mënyrë të qartë për një aplikim janë mohuar “by default”. Si “default”, aplikacionet nuk mund të aksesojnë direkt aplikimet e tjera ose hapësirën e OS.

Çdo aplikim Android vepron i izoluar në hapësirën e tij të përdoruesit në krye të një makine virtuale. Teknologjia Dalvik lejon pajisjet mobile të drejtojnë disa makina virtuale në të njëjtën kohë në mënyrë efikase. Dalvik VM mbështetet në kernel për themelimin e funksionaliteteve të tilla si fillesë/threading dhe menaxhimin e kujtesës të nivelit të ulët. Frameworku i Aplikimit është krijuar nga kodi i hartuar për makinën virtuale Dalvik, dhe është përgjegjëse për shërbime të ndryshme dhe komponente që janë përdorur nga aplikime të ndryshme që ekzekutohen në krye të

tyre. Në shtresën më të lartë janë aplikime kyç/të rëndësishme që janë para-instaluar nga prodhuesi dhe ato që mund të shtohen nga përdoruesi.

Edhe pse Android është ndërtuar në majë të kernelit Linux ai mbështetet në kernel për nënsistemet e tij të ndryshme OS sikur *Scheduler, menaxhimin e kujtesës, zbatimin e procesit, driverat e pajisjes, mbështetje rrjeteve etj*, ajo shton zgjerimet e veta (own enhancements) për kernelin kryesor (mainline). Kështu që edhe pse disa botime kërkimore pretendojnë se Android vepron si një sistem operimi në vetvete, por duhet thënë se ai - Android është vetëm një kornizë/frameworku software ndërtuar në majë të kernelit Linux.

7.1.2 Makina Virtuale Dalvavik

Pajisjet mobile kanë disa kufizime si në procesor, kujtesën apo baterinë krahasuar me pajisjet e tjera portabël. Dalvik VM [90] është një makinë virtuale e projektuar dhe optimizuar për mjedise me kapacitet të ulët si pajisjet e lëvizshme. Çdo aplikim Android ekzekutohet brenda një instance të Dalvik VM dhe shumë instanca mund të bëhen njëkohësisht dhe kanë një ndikim shumë të ulët në performancën e kujtesës së pajisjes smart. *Dalvik VM përdor fila .dex për të ekzekutuar aplikacionet. Një skedë .dex mund të kombinonojë disa fila .class, e cila i lejon ruajtje më të efektshme.* Mjeti/toolsi "dx" i përfshirë në Android Studio / SDK mund të transformojë klasat e kompiluara (*.class*) nga një kompilues/ përpilues i gjuhës Java në fila *.dex.*

7.1.3 Aplikimet Android

Aplikacionet Android sikurse përmendëm janë të bazuara në filën e paketës Android (.apk), e cila është skedari i instalimit i krijuar nga përpilimi/përkthimi i kodit Java të aplikimit duke përdorur SDK Android dhe një ambient të zhvillimit JAVA tilla si Eclipse [122]. *Një file .apk është një koleksion i komponentëve që ndajnë një grup burimesh.* Aktiviteti: ai përmban ndërfaqen e përdoruesit. Komponentët e aktivitetit mund të konsiderohen si shtresë prezantimi për aplikimet.

Aplikacionet Android janë shkruar në Java dhe ato përbëhen nga komponentë të ndryshëm me funksione të ndryshme. Në një kohë të caktuar vetëm një aktivitet në sistem mund të ketë vëmendjen dhe fokusin e procesimeve duke mbajtur gjithë aktivitetet e tjera në gjendje pezull. Aktivitetet mund të ekzistojnë në tre gjendje:

-*Resumed* ose *running*. Aktiviteti është në *foreground (veprimtari aktive)* në ekran dhe ka fokusin e përdoruesit.

-*Paused*. Një aktivitet tjetër ka fokusin, por ky i pari është ende i shikueshëm nga sistemi. Një aktivitet i bërë "pause" është tërësisht "në jetë / punë" dhe objekti i tij është mbajtur në memorie.

-Aktiviteti është tërësisht "i fshehur" nga një aktivitet (*background*). Aplikimet janë të ngarkuara në sfond dhe nuk kanë vëmendjen e përdoruesit.

7.1.3.1 Aplikimet dhe periodiciteti

Ndërkohë, versione të shumë aplikimeve si ai Facebook apo Email kanë një mënyrë operimi të ndryshme nga disa të tjera si p.sh kjo në lidhje me mënyrat e updateve duke qënë aktivë dhe në sfond. Rezultatet kanë treguar se këto aplikacione kanë një periodicitet për updatet dhe periudhat

kohore të tyre mund të rregullohen në “settings” të cdo aplikimi.. Pyetja kryesore që ne shpesh kemi shtruar është se nëse sistemi Android i ben “align” këto sinkronizime aplikacionesh, dmth, nëse aplikimet janë sinkronizuar me kohë të ndryshme në fillim, nëse ata janë sinkronizuar njëkohësisht pas disa provave, ose duke u bërë gradualisht më të afërt dhe më në fund identike. Ne realitetin e sotëm të Android, mund të bllokohen ose të lihen të lira në grup (me përzgjedhje të aplikimeve) ose të gjitha ndërveprimet background / sfond të aplikimeve c’ka nënkupton se ka një profil të sjelljes së OS që specifikisht përcakton këto kufizime.

7.1.3.2 Si i menaxhon Android aplikimet

Shumë aplikime mund të ekzekutohen në të njëjtën kohë. Vetëm një kërkesë e një aplikimi mund të ekzekutohet në një kohë. Pas mbylljes së aplikimit ajo përsëri qëndron në memorie. Aplikimet janë të shkruara në Java dhe të hartuara në kodin Dalvavik *(.dex*). Por pyetja përse me Dalvavik ?: Sepse ai optimizon më shumë memorien e brendshme (p.sh një file *.jar* 113 KB kompilohet në një file *.Dex* në rreth 20 KB), duke përdorur librari Java të grupeve të 3-ta.

7.1.4 Staku i rrjetit ne OS Android

Staku i SW të Android përfshin një *sistem operativ OS*, *middleware* dhe *një grup aplikimesh* kyç. OS i Android bazohet në versionin 2.6 të Linux, i cili jep shërbime bërthamë/qëndrore të sistemit dhe vepron si një shtresë abstrakte midis HW dhe pjesës tjetër të stakut të SW.

Një skemë e thjeshtuar e SW të stack-ut të Android jepet si në figurën 7.2. Sikurse kemi thënë shumica e aplikimeve të Android janë shkruar në gjuhën Java dhe ekzekutohen në një mjedis Java Dalvavik (JRE), i cili ka si qëllim bazë zhvillimin dhe ekzekutimin e aplikimeve. Androidi përfshin një set/grup të librarive të veta, të cilat japin shumicën e funksionaliteteve sikurse edicioni standard i Java (Java SE).

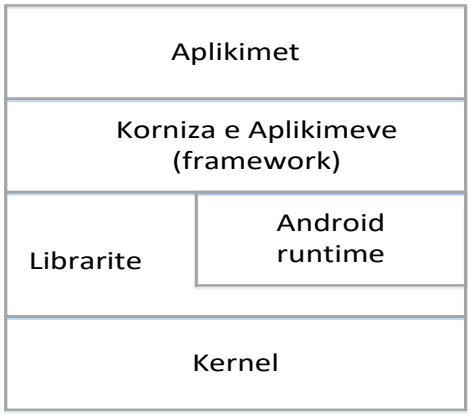


Figura 7.2: Staku i thjeshtuar i Android OS

Normalisht çdo Aplikim Androdi kryen (run) proceset e veta të cilat shihen si kërkesa të makinës virtuale Dalvavik. Dalvavik është optimizuar për të egzekutuar shumë kërkesa të njëkohshme në makinën virtuale. Aplikimet janë të “paketuara” në një format special të egzekutueshëm Dalvavik *.dex* (dhe jo *.jar* sikur janë në Java SE), i cili është optimizuar për ndikim në memorie të pajisjes. Dalvavik mbështetet në Linux për funksionalitete në nivel të ulët (low level), për shembull menaxhimi në nivel të ulët i memories dhe mbushja e të dhënave. Struktura e Linux është e bazuar

në modelin TCP/IP, e cila në thelb përcakton 4 shtresa: *Aplikacion, Transport, Internet* dhe *Linkun e të dhënave*. Figura 7.3 tregon se si ky model komunikimi midis shtresave zbatohet dhe në Linux.

Hapësira e përdoruesit (API) ndërvepron me shtresat e ulëta përmes "thirrjeve" (calls) të sistemit. Këto thirrje të sistemit i lejojnë shtresës së aplikimit të vendosin komunikimin midis kernelit dhe hapësirës së përdoruesit me anë të "sockets" apo grepave. Ka tre lloj "sockets" / grepash kryesore në Linux:

•***socket te Netlink*** [116]: Ai përdor Protokollin e familjes AF Netlink.
•***socket te Unix*** Domain [134]: Ai përdor protokollin e familjes AF UNIX.
•***socket te Internetit***: Ai përdor familjen protokollit AF INET.

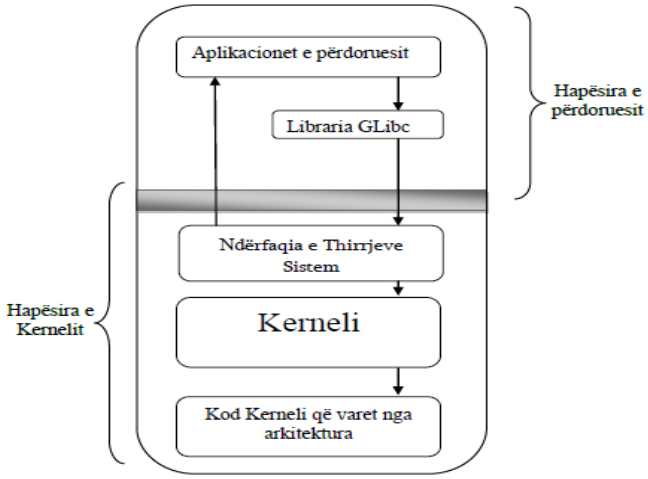


Figura 7.3: Arkitektura bazë e GNU/Linux

Android ofron një API të rrjetit [2] për të krijuar socket në domain UNIX. Për më tepër, thirrjet e reja të sistemit mund të përcaktohen në një ndërfaqe të gjuhës Java (JNI) për të mbështetur llojet e tjera të socket. JNI [90] është një kornizë/framework programimi që lejon aplikacionet Android JAVA për të thirrur programe amtare dhe bibloteka të shkruara në gjuhë të tjera programimi si C e C++.

Shtresa e Transportit është shtresa më e lartë brenda hapësirës Kernel. Ai përdor socket për të kaluar paketat e rrjetit nga hapësira Kernel në hapësirën e përdoruesit dhe anasjelltas. **Shtresa e Internetit** është përgjegjëse për verifikimin dhe rrugëzimin e paketave hyrëse dhe dalëse nga shtresat më të ulëta për te më të lartat dhe anasjelltas. Së fundmi, një protokoll i **shtresës së linkut** është përdorur për të dërguar dhe marrë trafik së pari në shtresën e tretë e cila është shumë e përdorshme për procesimin e paketave. *Informacioni i një pakete kopjohet në një strukturë buffer të Socket (sk_buff), e cila është zhvendosur nga shtresa në shtresë. Për më tepër, ajo kopjon informacionin e paketave të marra nga ndërfaqja e rrjetit në sk_buff në dhe i dërgon ato në shtresën e internetit. Paketat e marra nga shtresa e internetit janë dërguar më pas me anë të kartës së rrjetit.* Por brenda strukturës së OS, cila është pjesa përgjëgjëse për procesimine e të dhënave

Netfilter *[129] është një framework brenda Kernelit Linux që u jep mundësi funksioneve të rrjetit si përkthim adresash (NAT) dhe interceptim e paketash dhe modifikimi. Kështu që, fluksi i trafikut mund të kontrollohet dhe modifikohet në përputhje me ndonjë algoritëm të aplikueshëm. Netfilter*

jep mundësinë të mbartë trafik rrjeti në hapësirën e përdoruesit duke kombinuar modulet e Netfilter, socket (grepat) Netlink [116] dhe aplikimin API të hapesirës së përdoruesit për pritjet/queue libnetfilter_queue [31] i cili është falas në internet.

7.1.4.1 Shtresa socket

Një Socket jep një kuptim për të komunikuar midis dy proceseve që qëndrojnë ose në të njëjtin nyjë ose në nyje të ndryshme. *Sistemet operative përdorin "socket" si një abstraksion për të lidhur dy procese.* Çdo socket mirëmban adresën e burimit, destinacionit dhe është përdorur nga proceset për të dërguar dhe marrë të dhëna duke përdorur socket/grepat API të ofruara nga sistemet operative të shtresëzuara.

Qëllimi i socket është të modifikojë socket API për të lejuar aplikimin, për të shprehur preferencat e tij komunikuese si bandwidth i lartë, vonesat e ulëta, elasticitetin e lidhjes etj. Ai e përdor këtë informacion për të zgjedhur ndërfaqen e duhur, rregulluar parametrat e rrjetit, ose dhe kombinuar shumë ndërfaqe. Ato nuk bëjnë për përdorimin e shumë ndërfaqeve sepse ato zgjedhin një nga ndërfaqet bazuar në preferencat e aplikimeve përpara së të stabilizojnë lidhjen.

7.1.4.2 Struktura e bufferit të Socket

Bufferi i socket është struktura e të dhënave më të rëndësishme në stakun e rrjetit të Linux sepse ai përmban të gjithë informacionin e lidhur me një rrjet paketë. *Kjo strukturë lejon një paketë të lëvizë nga një shtresë në një shtresë tjetër derisa të arrijë në destinacion*.

Të gjitha pritjet /rradhët të lidhura me rrjetin në zonën kernel përdorin një strukturë të përbashkët të të dhënave: struct *sk_buff. Kjo është një strukturë (struct) e madhe që përmban gjithë informacionin e kontrollit të kërkuar për paketat (datagram, qeilze, çfarëdo).* Elementët sk_buff janë të organizuara si një listë e lidhur dyfishe, në mënyrë të tillë që ajo të jetë shumë efikase për të lëvizur një element sk_buff nga fillimi / fundi i listës në fillim / fund të një tjetër liste. Një radhë është përcaktuar nga sk_buff_head struct, e cila përfshin një kokë dhe një tregues bisht të elementeve sk_buff [138].

Të gjitha strukturat e queuing përfshijnë një sk_buff_head që përfaqëson pritjet / radhët. Për shembull, struct_sock përfshin një pranim dhe dërgim te pritjes / radhë. Ka funksionet për të menaxhuar radhët e gjata (skb_queue_head (), skb_queue_tail (), skb_dequeue (), skb_dequeue_tail ()) dhe të veprojë në një sk_buff_head. Në realitet sk_buff_head është përfshirë në listën e lidhur dyfishe te sk_buffs (kështu që në fakt formon një unazë). Kur një sk_buff është rezervuar, edhe hapësira e saj e të dhënave është rezervuar nga memorja e kernel. Rezervimi i sk_buff është bërë me alloc_skb () ose dev_alloc_skb (); driverat përdorin dev_alloc_skb (), falas nga kfree_skb () dhe dev_kfree_skb (). Megjithatë, sk_buff siguron një shtesë të shtresës së menaxhimit. Hapësira e të dhënave është e ndarë në një zonë kryesore dhe një zonë të dhënave. Kjo lejon funksionet e kernelit të rezervojnë hapësirë për kokën, në mënyrë që të dhënat të mos kopjohen vërdallë. Pra, pas rezervimit të një sk_buff, hapësira kokës / header është e rezervuar duke përdorur skb_reserve () .skb_pull (int len) - heq të dhënat nga fillimi i një buferi /memorjeje (duke lënë mënjanë një kokë ekzistuese) duke avancuar të dhënat te të dhënat+len dhe duke ulur "len".

Struktura sk_buff ka fusha që të tregojnë për koka(headers) specifike të shtresës së rrjetit:

- *transport_header* (i quajtur më parë h) - për shtresën e 4 (sipas modelit OSI), shtresa e transportit (mund të përfshijë header TCP, header UDP ose header ICMP, dhe më shumë).
- *network_header* - (e quajtur më parë nh) për shtresën e 3, shtresa e rrjetit (mund të përfshijë koka / header IP ose kokë / header IPv6 ose header arp).
- *mac_header* - (e quajtur më parë mac) për shtresën e 2, shtresa Link.
- *skb_network_header* (SKB), skb_transport_header (SKB) dhe skb_mac_header (SKB) kthejnë një tregues/pointer për header-in / kokën.

Figura 7.4 tregon fushat e *sk_buff* :

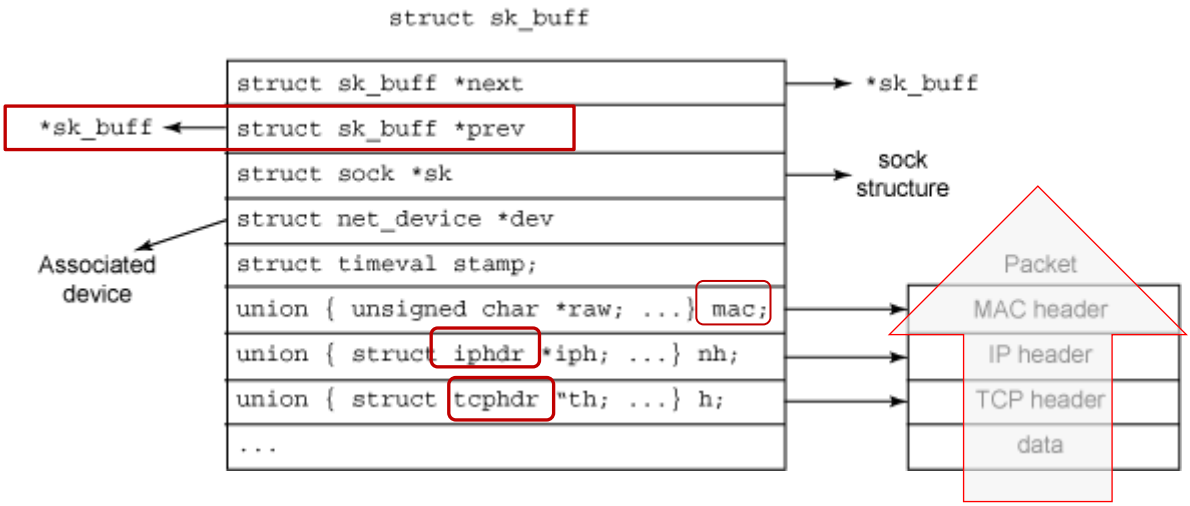


Figura 7.4: Struktura e memories të socket (*sk_buff*) [138]

Pra kjo strukturë përmban shumë tregues (pointera) të kokave të protokolleve të shtresave të ndryshme për të ndërtuar paketat hyrëse e dalëse. Për më shumë, ai ruan kohët e ardhjes dhe dërgimit dhe pajisjen në bashkëveprim nga e cila është marrë ose do të dërgohet.

7.1.5 Menaxhimi i fuqisë nga Android

Një nga ndryshimet që Google i bëri kernelit origjinal të Linux ishte shtimi i driverit të fuqisë të Android në kernelin ver 2.6.33. Driveri i ri është shtuar duke pasur parasysh pajisjet portabël që punojnë në OS Android të cilat kanë të kufizuar kapacitetin e baterisë dhe kanë siguruar shërbime të reja (featura) për kursim të energjisë që janë të ndryshëm nga ato të kompjuterëve personal.

Në shtresën/kornizën Aplikative një *API Power Management* është zhvilluar siç tregohet në figurën 7.5. Aplikacionet Android kërkojnë burimet e CPUve duke përdorur mekanizmat e quajtur zgjimi i kyçjeve (*wake locks)* përmes Kornizës/frameworkut të Aplikimit, Librarive dhe shtresës së Kernelit Linux. *Wake Locks* janë shërbime të manaxhimit të fuqisë që përdoren për të kontrolluar gjendjen e energjisë të pajisjes mbajtëse. Një *wake lock* që konsiderohet në gjendje të mbyllur (locked) pengon sistemin nga aksesimi në statusin pezull/suspend ose në gjendje në nivel të ulët të energjisë, në përputhje me llojin e saj.

Siç ilustrohet në Figurën 7.5, kur aplikimi Aplikimi (A, B ose C) ka nevojë për burimet e CPU, ajo dërgon një kërkesë te menaxhuesi i energjisë API i cili transferon kërkesën te driveri i energjisë i vendosur në shtresën Kernel Linux. Kur "wake-lock" është krijuar menaxheri i energjisë raporton incidentin përsëri mbrapsht në App1. Në përputhje me llojin e *wake-lockut* burimet e duhura janë konsumuar.

Përsëri kjo nuk e ka zgjidhur problematikën me konsumin e shpejt të energjisë në lidhje se si përdoruesi, aplikimet apo rrjeti me parametrat e tij ndërveprojnë në momente dhe kushte të ndryshme. *Zgjidhjet aplikative janë te kushtueshme nëse do të implementoheshin në shtresën aplikative pasi do të rriste shumë punën e CPUs (sepse përsëri API do të ndërvepronte me libraritë kernel për përmirësimin energjitik) dhe për rrjedhojë rezulton në një rritje të energjisë.*

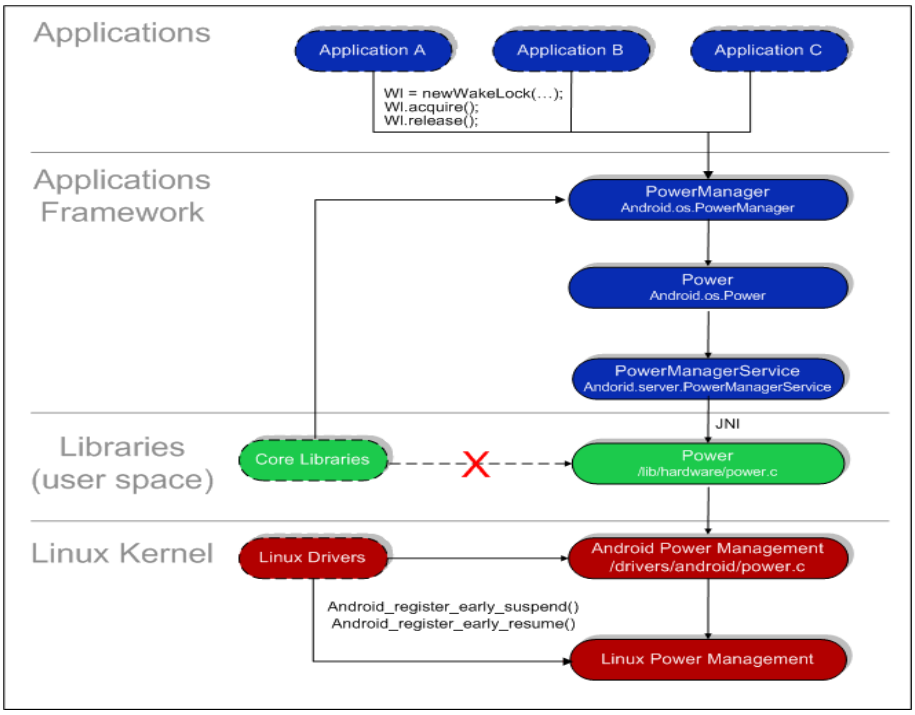


Figura 7.5: Arkitektura e menaxhimit të fuqisë në Android [90]

7.1.6 Programe zhvillimi në Android

Ka një numër të madh programesh/tool-se që duhen për të zhvilluar aplikimet Android. **Android SDK** (platformë zhvillues SW) është një strukturë modulare dhe përmban e jep paketa duke përfshirë tools-et e platformës për zhvillim dhe kontrollin për një aplikim. Mjedisi zhvillues i integruar Eklips (IDE) është përdorur gjatë zhvillimit si një tool që bën procesin e zhvillimit me efiçent. Tools-i i zhvillimit Android (**ADT**) është integruar në Eklips për të zgjeruar funksionalitetet e Eklips, dhe të japë funksione specifike IDE për zhvilimin në Android. ADT bën detyrat "rutinë" të tilla si fillimi i krijimit të një projekti në Android, krijimi i një ndërfaqeje aplikimi përdoruesi, kontrollin e aplikimeve duke përdorur tools-et e SDK Android më lehtë [90].

Tools-i logjik është një mekanizëm për mbledhjen dhe shikimin e të dhënave në dalje të kontrollit të sistemit. Të dhënat bazë në dalje të funksioneve në Java dhe shkruhen në toolsin Logcat. Tools-i Logcat është integruar me ndihmën e një shtesë ADT në Eklipsin IDE apo tashme si pjesë e

Android studio SDK, dhe është tools-i kryesor për kontrolle e debug. **ADB** është përdorur si një tools që jep një qasje të thjeshtë për të operuar filat e sistemit të smartphoneve dhe suportojë downloadim/shkarkimin e filave nga Smartphone, tashmë dhe pjesë e Android Studio. **DDMS** apo serveri monitorues i kontrollit Davlavik është një tools i përdorur për të bërë ose kapur fragmente (screenshots) për raportin final dhe të gjejë rrjedhjet e memories. Megjithatë në versionet e fundit të Android Studio, Eclipse nuk përdoret më pasi funksionet e tij përfshihen brenda vetë Android Studios. Të gjitha toolset e tjera gjithashtu janë tashmë pjesë e Android Studios kur ndër versionet stabël është ai 2.2.3.

7.1.7 Sfidat e Android, Portimi dhe AOSP

Edhe pse Android është standardi i duhur për teknologjinë e procesorëve ARM, portat komerciale të Android janë sot të mundshme në platformat si PowerPC, SH4 (Reneasas), MIPS, dhe x86. Ky portabilitet e bën sistemin operues mobile më të përdorshmin midis të gjithëve. Një njohje e mirë ku të gjejmë modulet e release-it te OS dhe / ose module të tjera poshtë faqeve web të Android Git Hub dhe faqeve web të prodhuesve të smartphoneve Android. Aplikimet duhet të krijohen përmes Android Studio duke përdorur versionet më të larta të saj. Por zgjidhjet në OS kërkojnë një njohje më të thelluar të arkitekturës së OS, toolse-ve dhe implementimit të zgjidhjeve.

Sikurse për shumë studiues dhe normalisht vendorë pajisjesh smartphone Android, sistemi OS Android është rritur me më shumë patche/zgjidhje dhe versione të fuqishme që kanë sjellë mbështetje për çdo lloj funksioni e kërkese. Për migrimin e sistemit Android te shumë pajisje smartphone (edhe për një modifikim procesi moduli SW) ne duhet të jemi të kujdesshëm për diferencat midis Linux normal dhe atij te Android. Politika e driver-ave të kernelit të Linux në Android është të shmangi modulet dhe ai ekzekutohet në një hardware me konfigurim fiks/specifik dhe jo njësoj sikurse Linuxi ku shpërndarja apo ekzekutimi i tij bëhet në pothuajse të gjithë hardware-t e mundshëm, kështu që ne duhet të jemi të sigurt që Kerneli është ai i duhuri për pajisjen UE nën testim dhe me versionin e Android-it.

Kerneli i Linux për Android duhet të hartohet në përputhje me hardware-in e pajisjes nën testim në të cilën ne duhet të ndërtojmë sistemin Android (pash në rastin e shtimit apo modifikimit të moduleve sw). Përveç Kernelit dhe driverave të UE, GUI i Android duhet të përshtatet me platformën e pajisjes nën testim.

Krijimi i "patcheve" (zgjidhjeve OS) në mënyrë që Kerneli Linux të jetë i harmonizuar për Android-in tashmë është bërë një zhvillim dhe zgjidhje e shpejt dhe e përsëritshme në kohë për pajisjet mobile dhe pajisje të tjera. Porsa të kemi Kernelin e Linux-it "të portuar", është shumë më e thjeshtë të marrim apo të kemi pjesën tjetër të sistemit Android në vend dhe pastaj ne mund të marrim avantazhin e një numri të madh aplikimesh Android që ekzistojnë në komunitete (web forume). Modulariteti nuk është përfshirë në dizenjimin e stakut kështu që paketat (imazh sw) dhe feature-at e paketës (imazhi sw) nuk mund të ndryshohen kollaj. Staku i Android ka një varësi të fortë me hardware-in e pajisjes UE (smartphone në rastin tonë). Procesi i modifikimit dhe shtimit të moduleve të rinj në një sw egzistues, merr emërtimin "portim".

Procesi i ***Portimit*** *është mundësia që të krijojmë një imazh sw specifik për një hardware specifik për të vendosur në gjendje pune një pajisje specifike (UE) duke përdorur makinën host dhe duke zhvilluar mjedise të përshtatshme për të hartuar/përkthyer dhe ndërtuar burimin (e imazhit sw) me ndihmën e platformave hibride (cross) tool-chain dhe si përfundim ngarkimin e këtyre imazheve në pajisjen hardware nën testim.* Figura 7.6 tregon hapat e portimit të Android OS në një pajisje UE smart me bord ARM.

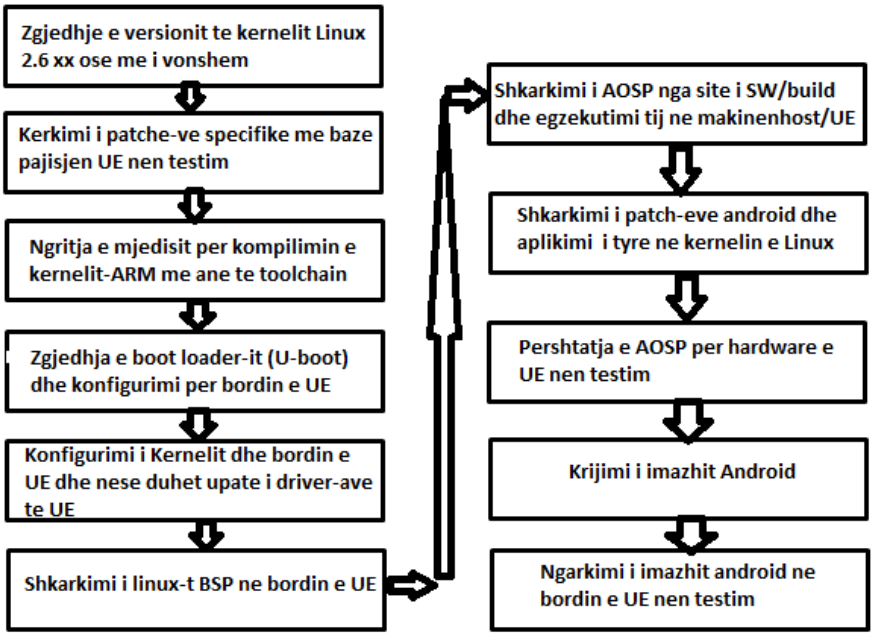


Figura 7.6: Hapat e krijimit/porting të imazhit Android OS në një board UE tip ARM [135]

Projekti përgjegjës për krijimin e sistemit Android është quajtur AOSP, dhe fillimisht është zhvilluar dhe mirëmbajtur nga kompania Google si mundësi e lirë (pa pagesë), dhe është një mundësi drejt hapësirave të reja për inovacione që shpesh nuk shihen në sisteme operative mobile OS's të tjera. Gjithashtu dhe si rrjedhojë e fleksibilitetit në përshtatjen dhe rregullimin e Android për lehtësira krijimi për më shumë hapësira të ROM's, opsione rooting të cilat mundësojnë shumë shërbime (feature) të reja. Pas sinkronizimit të AOSP nga website zyrtar ne mund të marrim nën-direktoritë SW sikurse shihet dhe në figurën 7.7 ku ndër nën-direktoritë më të rëndësishme janë:

```
sit@sit-desktop:~/home/Myandroid$ ls
bionic     cts          device      hardware  ndk       sdk
bootable   dalvik       external    libcore   packages  system
build      development  frameworks  Makefile  prebuilt
sit@sit-desktop:~/home/Myandroid$
```

Figura 7.7: Nën-direktoritë sw në direktorinë e AOSP

/bootable/ - kodet për procesin e boot dhe të ngritjes (HW). */dalvik/* - kjo direktori përmban makinën virtuale Dalvik në */dalvik/vm*, përpiluesit .dex në */dalvik/dx* dhe bibliotekave të klasave të bërthamës, si dhe mjete të ngjashme. */framework/* - direktoria që përmban kornizën Android për të mbështetur të gjithë sistemin dhe aplikacionet e përdoruesit. */ndk/* - kjo ndk ka një grup mjetesh për ndërtimin kodin amtar që mund të jetë kod gjurmues për në memorien e ulët, kodi CPU-

intensive dhe përpilimi/ përkthimi për të kaluar hartimin e kodin tonë për arkitekturën e pajisjes të synuar. */system/* -kjo është një direktori me shumë gjëra të lidhura me sistemin, si dhe ruan kodin burim për bërthamën e sistemit bazë Android. Kjo gjithashtu përfshin kodin burim për procesin e initzializimit (ngritjes) dhe skriptin *init.rc* që mundëson konfigurimin dinamik të platformës.

Eshtë e rëndësishme të shikohet kompatibiliteti (përputhshmëria) përfshirë memorien, ruajtjen, rezolucionin e ekranit, kërkesat e grafikës, kërkesat e specifikimeve që një SW "i ri" duhet të kënaqë dhe një pajisje mobile duhet të mundësojë kompatibilitet të aplikimeve Android, duhet të mundësojë suport në fushën e aplikimeve online, duhet të mundësojë shërbime pa modifikuar ndonjë nga mundësi/shërbimet e Frameworkut dhe Google APIs.

7.2 Implementimi i algoritmit në OS Android

Ky seksion jep detajet e një zbatimi Android të algoritmit të përzgjedhjes së transmetimeve të të dhënave në kanale në bazë të konsumit të energjisë për secilin si një pjesë e nivelit të Kernel (KLS) në platformën Android. Për zbatimin e modulit software ne kemi marrë ndihmën dhe të inxhinierëve cilët kanë njohuri të theksuara në fushën e programimit në gjuhët C / C++ dhe ndërveprimin me kernelin e Linux. Gjithashtu në ndihmë na kanë ardhur dhe shumë diskutime në forume apo dhe shembuj imlementimesh të ngjashme për nga llogjika. Llogjika e funksionimit / veprimit të modulit implementohet në hapësirën e kernelit përderisa "throughputi" në hapësirën kernel është më i lartë dhe një aplikim i hapësirës së përdoruesit do të kërkojë bashkëveprim me hapësirën kernel përmes thirrjeve të sistemit. Kjo do të krijonte një aktivitet pak më të lartë të CPU për shkak të transferimit të të dhënave (lexim / shkrim) midis shtresave, p.sh., një transferim i vetëm i të dhënave mund të rezultojë në një shfrytëzimin të CPU deri në 33% sipas autorëve te [114]. Zbatimi i zgjidhjes krijon një modul/file sw të ngarkueshëm (*lkm*) kernel brenda strukturës së netfilter që mund të futet / portohet në kohën e duhur për testime (në vend të ndryshojmë kodin amtar në kernel në kohën e hartimit/krijimit).

Netfilter është një kornizë/framework kerneli Linuxi që mundëson "përgjimin" e paketave dhe modifikon grepat e ndryshëm (pikë vendimtare në stakun e rrjetit). Llogjika në modul regjistron grepat netfilter NF_IP_POST në shtresën e IP të protokoll stakut TCP/IP. Kjo na lejon për të kapur paketat dalëse pa bashkëvepruar me aplikimet, duke çuar në një ndërveprim të ulët të CPUs. Implementimi është vendosur në një pajisje "të rrugëzuar" (rooted) Android (Samsug S3/4 apo me i ri si model). Ne ndër-përpilojmë kernelin Linux me opsionet e nevojshme (psh, të mundësojë mbështetje të ngarkimit të modulit dhe Netfilter), boot-ojmë Kernel-line ri në pajisje, kryejmë ndër-përpilim të modulit tonë dhe e ngarkojmë në sistem në kohën kur duhet të kryhen testime.

Duke analizuar dhe njëherë se, protokolli i kontrolluesit të radio burimeve (RRC) në mënyrë drastike influencon energjinë për bit në një transmetim, psh. një mesazh teksti i vogël mund të konsumojë më shumë se një video-konferencë në përdoruesin fundor. Duke kryer matjet në transmetimin e energjisë të pajisjes fundore ne tregojmë se parametrat "e fshehura" të rrjetit si kohëzuesit e pasivitetit për kyçje midis kanaleve, ose kufijtë e të dhënave buffer RLC mund të përdoren për të caktuar/planifikuar transmetimin e paketave në një mënyrë efiçente të energjisë.

Në përmbledhje, puna jonë është së pari në shqyrtimin e matjen e të dy ndikuesve; energjisë bisht të transmetimit dhe volumit të të dhënave të bufferit RLC në kombinim me planifikimin e transfertave të të dhënave. Është treguar se bishti (Tail) si dhe të dhënat e bufferave RLC mund të optimizohen dhe kështu të kemi më pak energji të konsumuar sesa pa ndonjë zgjidhje të këtij lloji. Për të arritur ketë duhet të përdorim njohuritë e shtresave te TCP/IP në mënyrë që të planifikojmë që të dhënat të transmetohen më efiçente për sa i përket energjisë. Një llogjikë që planifikon transmetimet e të dhënave në sfond bazuar në parametrat e lartpërmendur të rrjetit sjell një kursim nga 20 deri në 40 % të energjisë për aplikimet e zakonshme SN si Facebook, Twiter, BBC News etj. Megjithatë problematika të reja mund të hasen si rrjedhojë e vonesave në llogjikën e modulit duke shkaktuar rritje të ritansmetimeve etj. Figura mëposhtë jep strukturën e përgjithshme të modulit të propozuar:

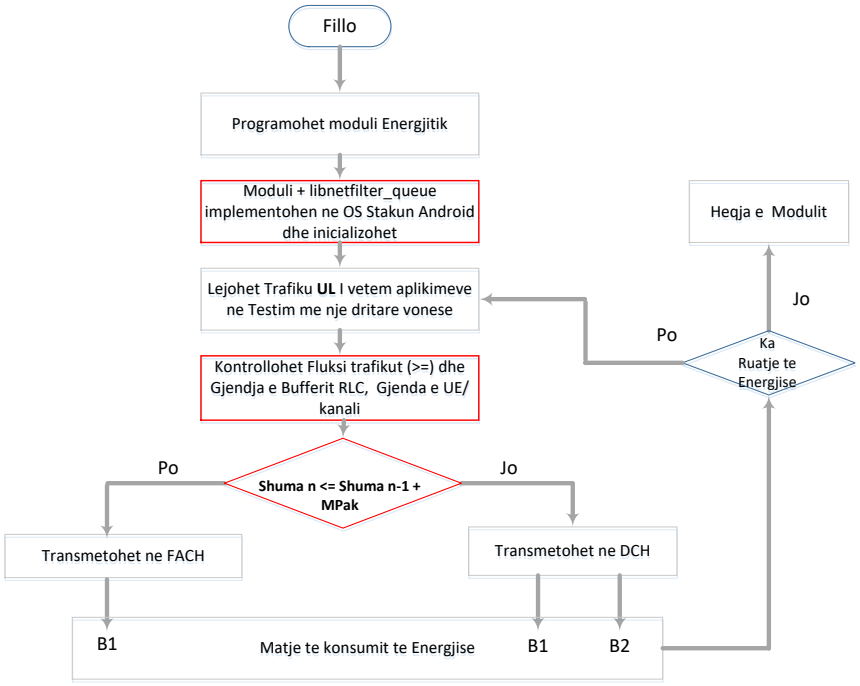


Figura 7.8: Bllok skema e përgjithshme e funksionimit të zgjidhjes së propozuar

Zgjidhja që ne propozojmë është përpilimi (programimi) i një moduli / file në C dhe vendosja e këtij moduli në kernelin e pajisjes smart Android duke u implementuar si një modul tërësisht i ri në shtresën e rrjetit. Ky proces njihet si "porting" dhe kërkon një ri-kompilim të SW të Android. Detyra e këtij moduli është analizimi i trafikut të aplikimeve (paketat) në lidhje me fluksin apo volumin e tyre dhe të bëjë një vlerësim të gjendjes së memorjeve të RLC (ku 2 buffera B_1 / P_V e B_2 / P_M janë krijuar për të mbajtur paketat e vëna në pritje) dhe statusit të kanalit në ndërfaqen e komunikimit të 3G/UMTS apo 4G/LTE. Bazuar në këto vlerësime si dhe në një kohë limit vonese Kv, vendoset se në ç'kanal komunikimi do të transmetohet dhe për rrjedhojë dhe transmetimi i të dhënave në buffer do të kryhet. Në figurën mësipër pjesa e shtuar (moduli sw kernel dhe funksionet shtesë) në OS Android përfshin kryesisht funksionet shtesë të implementimit fillestar dhe kontrollit të fluksit përmbi funksionet normale që ka apo kryen vetë sistemi operativ Android.

Në lidhje me problematikat e transmetimeve TCP të përmendura me parë, disa parametra të TCP (ritransmetim, dublikim ACK dhe jashtë radhe) duhet të modifikohen në kernelin Android (nga vlerat default) për të reduktuar deri diku këto probleme. Pashmangshmërisht krijimi i vonesave në transmetim mund të sjell dhe probleme të fluksit të trafikut që përdorin protokollin TCP për të cilat do të duhet një vëmendje më e madhe në të ardhmen. Ndërsa komunikimet UDP nuk preken nga këto problematika.

7.2.1 Menaxhimi i trafikut për të dhënat

Transmetimi i paketave të rrjetit në përputhje me një algoritëm për ruajtje të energjisë kërkon trajtimin e trafikut të rrjetit. Ne do të shpjegojmë alternativat kryesore të dedektimit të trafikut dhe modifikimin e trafikut të rrjetit brenda kernelit Linux dhe pjesës së përdoruesit. Ndër alternativat kryesore për të reduktuar dhe modifikuar trafikun brenda kernelit Linux janë: implementimi si një modul i ri Netfilter dhe modifikimi i Kernelit (filave ekzistuese te sistemit).

7.2.1.1 Bazat për modul kerneli Linux (.ko)

Kerneli i Linux-it konsiderohet si misterioz dhe i vështirë për shumë programues, me disa rregulla të veçanta dhe praktika programimi. Modulet e kernelit Linux janë një pjesë e softuerit që mund të ngarkohet / shkarkohet sipas kërkesës, qoftë në fillim të ngarkimit të sistemit, ose në mënyrë dinamike kur sistemi është në funksionim. Modulet kernel zgjerojnë funksionalitetet e kernelit Linux pa ri-hartimin dhe ri-startimin e sistemit Linux. Shumë nga driver-at e Linux-it janë shkruar si module kernel dhe janë ngarkuar kur zbulohet një pajisje e përshtatshme për tu ngarkuar. Moduli i ri kernel në linux ekziston si file apo modul .ko në sistemin e skedarëve të Linux.

Linux ka disa grupe komandash cli për ngarkimin, shkarkimin dhe listimin e moduleve të kernelit:
1. *lsmod*: listoni një listë të moduleve kernel, madhësive të tyre dhe moduleve që varen prej tyre.
2. *modinfo <emri i modulit kernel ose .ko file name>:* shfaq informacion rreth një moduli kernel dhe punon për programin e ngarkuar kernel dhe skedarin .ko.

3. *insmod <.ko file name> dhe rmmod <emri i modulit kernel ose .ko file name>:* ngarkon një modul kerneli linux dhe mund të hiqni një modul kerneli të ngarkuar gjithashtu.
4. *modprobe:* ngarkon dhe shkarkon module nga kerneli Linux. Gjendet në dosjen e moduleve */lib/modules/ $ (uname -r)* për të gjitha modulet dhe skedarët e tjerë.

7.2.1.2 Implementimi në kernel apo në pjesën e përdoruesit

Ka një rëndesi të vecantë se ku mund të implementohet zgjidhja dhe këto janë shtresat e platformës Android ku mund të implementohet lehtësisht. Nga analiza të ndryshme ka disa mundësi por me kryesoret mund të rendisim dy alternativa: si ajo e zonës kernel dhe ajo zonës së përdoruesit. Implementimi në hapësirën e përdoruesit (user space) të lejon zgjidhje më të mëdha si rrjedhojë e mundësive të memories si dhe është dhe më e sigurt sesa implementimi në hapesirën kernel. Implementimi në kernel ka disa të meta si psh ndonjë problem apo gabim në programim, do të kemi një ndikim direkt në funksionet e sistemit operues apo mbi funksionet bazë. Edhe pse zgjidhjet aplikative kanë disa përparësi përsëri ato duhet të ndërveprojnë me kernelin përmes thirrjeve të sistemit për të bërë analizim. Realisht ky proces paraqet një shtesë në funksionet e

CPUs sepse të dhënat duhet të kopjohen nga kerneli te shtresa e përdoruesit (user space) dhe anasjelltas.

Nga disa studime nga autoret te [41] pasi mblodhen të dhëna për kostot ekstra të CPUs shkaktuar nga kryerja e thirrjeve të ndryshme të rrjetit në implementimin e funksioneve të modulit për trajtimin e paketave të rrjetit, arritën në përfundimin si në tabelën mëposhtë ku sic shihet kopjimi i të dhënave në bufferin e kernelit është pjesa me intesive e operimit të CPUs, me një rritje deri në 33% në përdorimin e punës së CPU.

Emri i funksionit	**Shtesa e punës së CPU-ve**
skb_copy_bits	33%
copy_to_user	3%
_alloc_skb	2.6%

Tabela 7.1: Shembuj kostoje të krijuar nga ndërveprimi kernel – user [41]

Pra zgjidhjet aplikative kërkojnë më shumë punë sepse kërkojnë më shumë ndërveprime me nivelet më të ulta pra kernelin, c'ka rit dhe kostot e punës së CPUs si dhe kosto energjitike.

Në anën tjetër, cungimi apo modifikimi i kernelit (egzistues), nuk është zgjidhje modulare sepse zgjidhja e trajtimit të paketave duhet të "integrohet" në kodin burim Kernel dhe në shumë module ndoshta. Për më shumë, është shumë e vështirë sepse në duhet të ri-kompilojmë të gjithë kernelin kur kodi burim është ndryshuar ose nëse do të ndryshohet më vonë. Megjithatë, metoda e cunngimit të kernelit mund të japë një throughput më të mirë të TCP/IP sesa zgjidhjet netfilter nëse zgjidhja është implementuar në rregull pasi jep një throughput më të mirë dhe mospërfshirje intesive apo shtesë të CPUs. Por ndryshime të kërkuara kërkojnë ndryshime në shumë fila dhe rikompilim të gjithë kernelit.

Edhe pse zgjidhja e modifikimit kernel mund të japë një throughput TCP/IP më të mirë m.q.s është në zemër të kernel, ne mund të përmbledhim të gjithë zgjidhjen në një modul të vetëm dhe për këtë kemi vendosur të implementojmë si një modul shtesë në netfilter.

7.2.2 Filtrimi i paketave te rrjetit me anë të grepave Netfilter

Netfilter është një grup "grepash" / hooks brenda kernelit të Linux. Kjo i lejon moduleve të Kernelit të regjistrojnë funksionet "callback" me stakun e rrjetit, në mënyrë që të kapin dhe manipulojnë paketat e rrjetit. Nje shembull konkret ka dhënë për krijimin e një firewall nëpërmjet Netfilter dhe jepet në weblinkun [146] dhe e cila ka qënë indicia për fillimin e modulit tonë. Gjithashtu dhe materialet në netfilter.org kanë dhënë një ndihmesë të madhe së bashku dhe me ndihmën e programuesve në këtë fushë.

7.2.2.1 Grepat e Netfilter

Paketat IPv4 që përshkrojnë strukturën e netfilter mund të ilustrohen si në figurën e mëposhtme ku dallohen shumë grepa kryesorë:
Kur një paketë rrjeti vjen në hyrje, ajo kalon nëpër grepin e parë NF_INET_PRE_ROUTING te Netfilter. Pas kësaj, paketa shkon përmes kodit të rrugëzimit, i cili vendos se për ku paketa është

e destinuar, ose një tjetër port në të njëjtën ndërfaqe rrjeti ose një ndërfaqe tjetër. Ajo gjithashtu mund të rrëzojë paketën nëse është e parrugëzueshme.

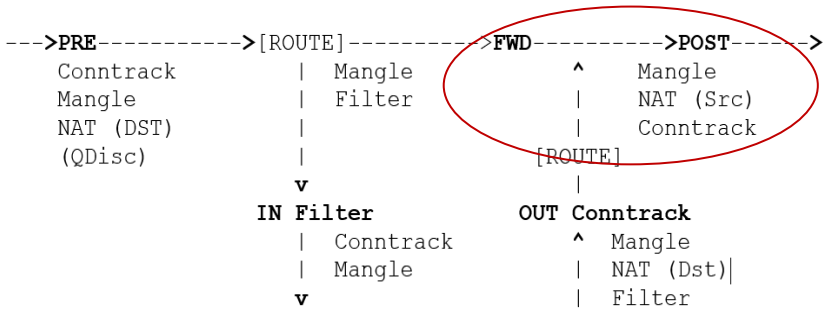


Figura 7.9.a: Struktura e trajtimit të paketave [135]

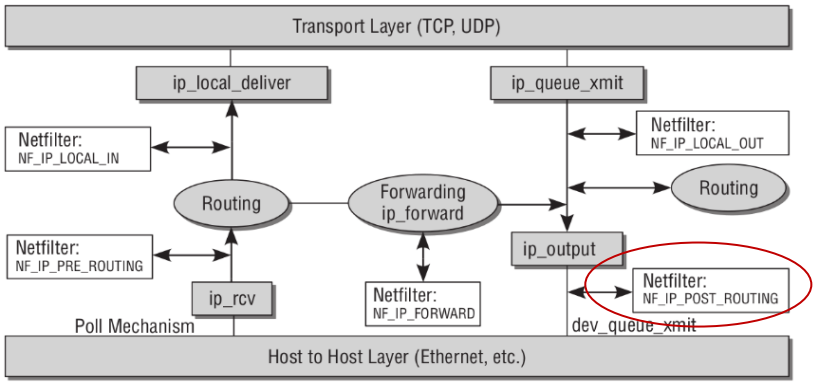


Figura 7.9.b : Struktura dhe Grepat e Sistemit Netfilter [129]

Nëse paketa shkon në një portë tjetër në të njëjtën ndërfaqe, grepi i dytë NF_INET_LOCAL_IN do të trigerohet. Kjo ndodh përpara se paketa të arrijë në portën destinacion. Nëse paketa shkon te një tjetër ndërfaqe e rrjetit, një grep i tretë NF_INET_FORWARD thirret, i ndjekur nga një grep i katërt NF_INET_POST_ROUTING përpara se paketa të arrijë mjedisin e transmetimit/daljen përsëri. Mbetet dhe një grep i fundit, NF_INET_LOCAL_OUT.

Ai thërritet për paketat e brendshme që dërgohen jashtë. Kodi i rrugëzimit thërritet përparara se ky grep të gjej adresën IP dhe më pas ky grep vendos të rrugëzojë. Modulet e kernel mund të rregjistrojnë secilën prej 5 grepave të përmendur [129].

Në figurën mëposhtë 7.10 pjesa me të kuqe tregon se ku mund të implementohet fila e re brenda netfilter për të bërë të mundur interceptimin dhe përpunimin sipas kushteve që në duam.

7.2.2.1 Implementimi i funksionit "Grep"

Porsa funksioni i grepit është rregjistruar me një nga 5 grepat Netfilter, ai do të thirret kur një paketë e rrjetit shkon përmes grepit në stakun e rrjetit. Duke rithirrur funksionin e regjistruar si më poshtë:

```
unsignedint hook_func_in(unsignedint hooknum,
struct sk_buff *skb,
conststruct net_device *in,
conststruct net_device *out,
int (*okfn)(struct sk_buff *))
```

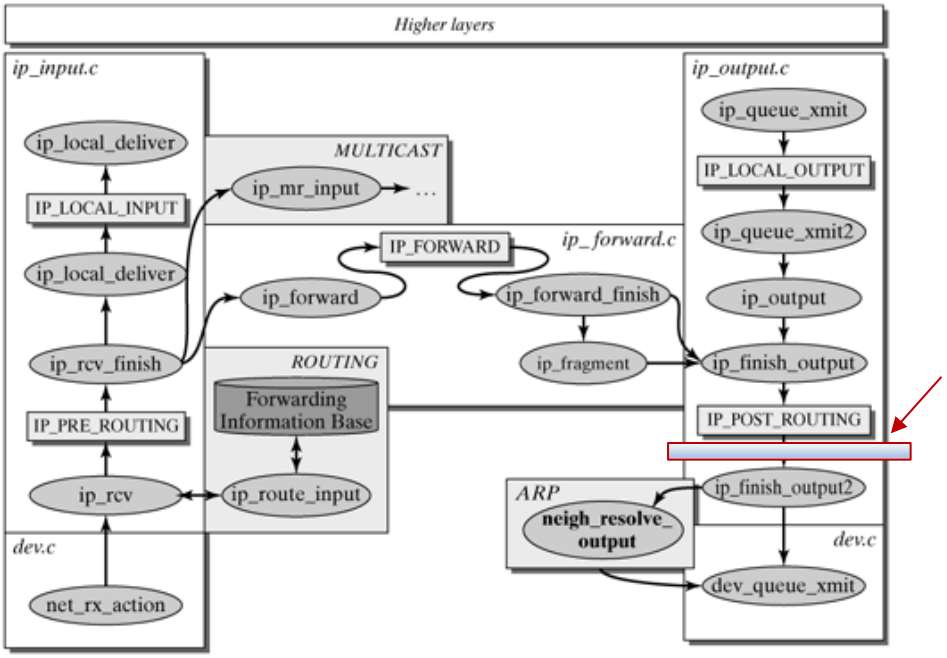


Figura 7.10: Mekanizmi i trajtimit të paketave IP në Linux, modulet përkatës [80]

hooknum tregon një nga 5 tipet e grepave (NF_INET_PRE_ROUTING, NF_INET_LOCAL_IN, NF_INET_FORWARD, NF_INET_LOCAL_OUT, NF_INET_POST_ROUTING) të treguara më sipër sikur në Figurën 7.9.a.

skb: është një tregues (pointer) në memorien (bufferin) e paketës së rrjetit, që është përcaktuar në modulin: */lib/modules/$(uname -r)/build/include/linux/skbuff.h*. skb në funksionin e grepit i mundëson një përdoruesit të përdorë memorjen/bufferin e paketës së rrjetit dhe të tërheqë lehtë informacionin e nevojshëm. Ndoshta tre informacionet më të kërkuara janë transport_header, network_header dhe mac_header fusha të përcaktuara në sk_buff [147].

In dhe *out* janë dy tregues të strukturës *net_device*, të cilat janë çfarë kerneli i Linux përdor për të përshkruar ndërfaqen e rrjetit, të përcaktuara në: */lib/modules/$(uname -r)/build/include/linux/ /netdevice.h.* Në funksionin e grepit, përshkruhet ndërfaqja e rrjetit që paketa i kalon përmes. Kështu që, në varësi të paketave të tërthorta, qoftë In ose Out do të jenë NULL.

Kur një paketë del pra në drejtimin UL, ai shkon nga shtresa e aplikimit, për te shtresa e transportit, pastaj shtresën e rrjetit, e kështu me radhë. Këto funksione normalisht ekzekutohen dhe skb_transport_header punon mirë. Të njejtat funksione të përcaktuara këtu përdoren dhe kur një paketë vjen nga rrjeti.

Okfn është një funksion tregues që lejon rregjistrimin e një funksioni "callback" të trigeruar kur të gjitha funksionet e rregjistruara me këtë grep kthejnë NF_ACCEPT. Kur një funksion i rregjistruar është thërritur, ai mund të kryejë një nga 5 gjërat dhe të kthejë një vlerë korresponduese, sikur është përcaktuar në *netfilter.h: NF_ACCEPT*: lejon paketën të kalojë për në pikën tjetër në stakun e rrjetit. *NF_DROP*: rrëzon paketën dhe gjithë burimet e lidhura me të. *NF_STOLEN*: Netfilter

“harron” rreth paketave dhe funksioni i grepit merr përgjegjësinë për të marrë paketën dhe nuk e lejon paketën të kalojë por e proceson atë. *NF_QUEUE*: vë në pritje paketën, zakonisht për përdorimin në hapësirën e përdoruesit dhe mund të riinjektohet/riimplementohen në stakun e rrjetit duke thirrur funksionin *nf_reinject()*. *NF_REPEAT:* thërret funksionin e grepit përsëri [136].

7.2.2.2 Procesi i kapjes dhe përpunimit të paketave

Në mënyrë që të interceptohen/kapen paketat dhe modifikohen gjithashtu, një grep duhet të rregjistrohet. Një grep / hook mund të rregjistrohet si një modul kernel i ngarkueshëm (*lkm*) i shkruar në gjuhën e programimit C duke përdorur funksionin e rregjistrimit *hook()*. Struktura për rregjistrimin e një grepi është *nf_hook_ops* dhe është e përcaktuar në modulin SW *linux/netfilter.h* i cili përmban këto fusha: *list:* kjo listë specifikon listën e lidhur ku strukturat *nf_hook_ops* janë ruajtur brenda Kernelit Linux. *hook():* ky është tregues për funksionin e lidhur me këtë grep. *pf(protocol family):* ky është një tregues për familjen e protokollit (PF_INET për IPv4). *hooknum:* kjo fushë tregon numrin e grepit. *priority:* kjo vlerë vendos prioritetin e funksionit të grepit.

Makro NF_HOOK lejon kodin e rrugëzimit për të përpunuar funksionet e lidhura me një grep. Kjo makro është përcaktuar në *linux/netfilter.h* dhe përmban argumentet e mëposhtme: *hook():*dhe *pf (protocol family):* sikurse përmendëm më sipër.*skb:* ky është një tregues për strukturën *sk_buff* që përmban të gjitha të dhënat e paketës së rrjetit. *inDev(input device):* ky është tregues për strukturën *net_device* dhe ajo tregon strukturën e ndërfaqes së rrjetit që ka marrë një paketë. (NULL për paketat dalëse). *outDev(pajisje output):* ky është një tregues për strukturën *net_device* dhe ajo tregon ndërfaqen e rrjetit përmes së cilës një paketë rrjeti duhet të lërë pajisjen. (NULL për paketa hyrëse). *okfn() (okay function):* ky është një funksion që thirret një herë kur funksionet e lidhur me një grep kthejnë vlerën NF_ACCEPT. Kjo do të thotë që kjo paketë është pranuar dhe mund të vazhdojë përshkimin e saj.

7.2.2.3 Menaxhimi i paketave IP brenda kernel Linux dhe hapësirës së përdoruesit

Në protokollin IP, ka tre faza kryesore në rrjedhën e një pakete në të gjithë shtresën e IP: *input*, *forward* dhe *output*. Kapja, menaxhimi dhe përpunimi i paketave përbëhet duke ndryshuar apo shtuar kodin në funksionet amtare/bazë Kernel të cilat ndikojnë në përshkrimin e një pakete në stakun e rrjetit. Këto funksione janë të shkruara në gjuhën e programimit C. Referuar figurës 7.11 për tre hallkat që merren me paketën në shtresën IP mund të themi se:

- Faza *“input”* merret me të gjitha paketat hyrëse dhe vendos nëse një paketë duhet të dorëzohet në nyjen/pajisjen lokale apo përcillet/kalojë më tej. Ajo zbatohet në filën/dosjen IP *input.c* dhe funksionet kryesore në kuadër të kësaj janë të shpjegura në shtojcën në fund.
- Faza *“forward”* proceson paketat që faza “input” i shënon si trafik forward (të kalojë më tej). Ajo zbatohet në dosjen e *ip_input.c* dhe funksionet kryesore brenda asaj janë në shtojcë.
- Faza e *“output”* vendos për të gjithë trafikun që po largohet. Ajo zbatohet në dosjen e *ip_output.c* dhe funksionet kryesore jepen në shtojcë:

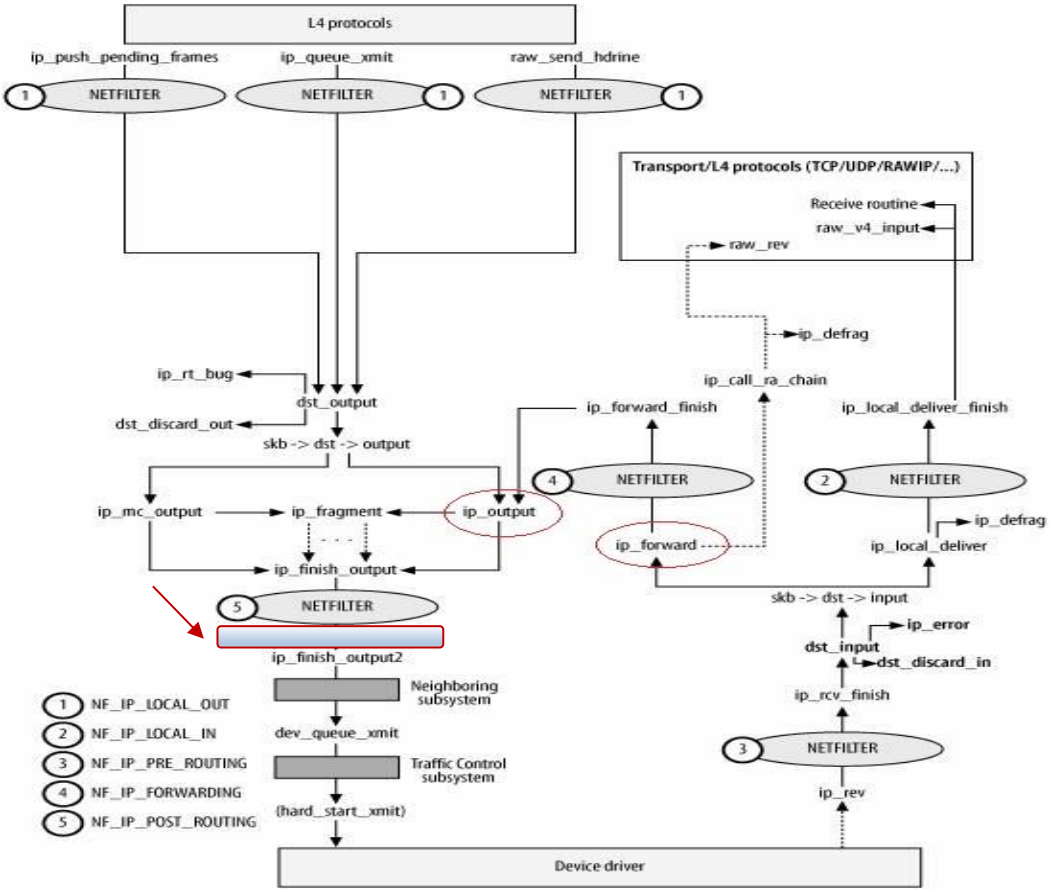


Figura 7.11: Fazat e përpunimit të paketave IP në shtresën e rrjetit në Linux [80]

Zgjidhja sikurse kuptohet konsiston në një filë të ngarkueshme në hapësirën e kernelit të Android, dhe është një filë e ndërtuar në gjuhën C, e cila ka dy funksione bazë: atë të inicializimit të modulit dhe të pastrimit të modulit nga funksionet. Kjo si filë duhet të vendoset pas grepit NF_IP_POST_ROUTING dhe para funksionit ip_finish_output2 për të kapur e ruajtur paketat në memorje e për ti transmetuar më pas në drejtimin UL.

Por netfilter na mundeson trajtimin e trafikut të rrjetit në hapësirën e përdoruesit duke kombinuar një modul Netfilter, grepat / socket's Netlink [116] dhe aplikimin *libnetfilter_queue* [81] të hapësirës së përdoruesit. Moduli apo fila e re në C duhet të kapë paketat në një grep të veçantë, dhe t'i vendosë në pritje ato te hapësira e përdoruesit (dhe kthen një vlerë NF_QUEUE në funksionin e grepit). "Sockets" të Netlink (të përcaktuara në *linux/netlink.h*) i transportojnë këto paketa nga kerneli te hapësira e përdoruesit. Tabela mëposhtë jep një përkufizim të Netlink.

Tipi	Arkitekturë portabël	Sinjalizim Event-Based	Lehtësisht i zgjerueshëm	Transferime të mëdha të dhënash
System Call	Jo	Jo	Jo	Po
/dev	Jo	Jo	Jo	Po
/proc	Po	Jo	Jo	Jo
Sysfs	Po	Po	Jo	Jo
Socket	Jo	Jo	Jo	Po
Netlink	Po	Po	Po	Po

Tabela 7.2: Specifikime të Netlink

Më pas, dy nga nga llojet e nevojshme të socket Netlink janë:

-NETLINK_ROUTE: ky opsion merr përditësimet e rrugëzimet dhe është përdorur për të modifikuar parametrat e rrugëzimit të tilla si tabelat e rrugëzimit apo adresat IP.
-NETLINK_FIREWALL: ky opsion transporton paketat IPv4 nga moduli kernel netfilter për te hapësira e përdoruesit.

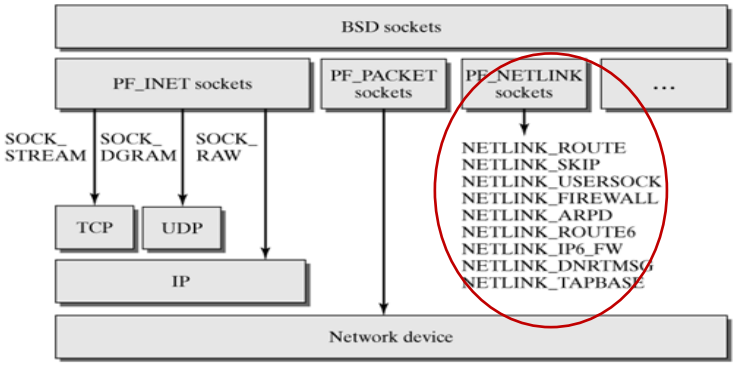


Figura 7.12: Grepat netlink në Linux

Më në fund, aplikimi *libnetfilter_queue* lejon hapësirën e përdoruesit të marrë paketat e vëna në pritje. Kështu që këto paketa mund të modifikohen dhe riimplementohen në stakun.

7.3 Implementimi i zgjidhjes në UE bazë Android

Në komunikimet mobile, ndërfaqja e rrjetit dhe parametrat e rrjetit që lidhen me to sikurse thamë janë ndër ndikuesit kryesorë në shpenzimin e shpejt të energjisë. Prandaj ne duhet të jemi të kujdesshëm për parametrat e rrjetit radio si p.sh kufiri i bufferit RLC apo dhe kohëzuesit e RRC. Kështu që, propozimi për një zgjidhje eficente ka të bëjë me atë që UE të reduktojë tranzitimet në DCH kur volumi i të dhënave nuk është vërtet shumë i lartë, dhe performon transmetimet e të dhënave në FACH (per llogjiken 3G) ose stopon në PCH për një periudhë kohe timeout pa transmetuar gjë (në llogjikën për 4G). Ky punim sikurse kemi thënë propozon një modul apo file shtesë netfilter për trajtimin e paketave të rrjetit te shtresa IP brenda hapësirës kernel në platformën e Android. Figura e mëposhtëme tregon një arkitekturë të përgjithshme të zgjidhjes për menaxhimin e transferimit të paketave në shtresën e 2 - 3 të kernelit.

Moduli ka në funksion primar dedektimin dhe vënien në pritje të të gjitha paketave përpara se ato të lënë pajisjen dhe i ri-injekton ato në stakun e rrjetit. Për thjeshtësinë e algoritmit, ne vendosëm të përdorim të njëjtin minimum të vonesës së dritares së dërgimit për të gjitha paketat (Kv = 60, 90 e 120 sek, por s'ka rëndësi madhësia) në mënyrë që të thjeshtohet implementimi. *Në algoritmin origjinal të komunikimit, pra atë cka zbaton OS i smartphone çdo paketë mund të ketë dritare të ndryshme minimumi të vonesave të dërgimit.* Duhet theksuar se ne përcaktojmë një kohë *Kv* si minimum i dërgimit të dritares së vonesës. *Implementimi i algorimit duhet thënë se jep një ruajtje të energjisë për transmetimet por me koston e rritjes së vonesave.* Ne duhet të marrim në konsideratë se kur koha *Kv* bëhet më e gjatë vonesat bëhen më të mëdha. Kështu që, ne duhet të gjejmë një të mesme midis kohës *Kv* dhe vonesave për të patur një ruajtje të energjisë dhe një QOS të pranueshme. Sikurse shihet, zgjidhja energjitike konsiston në një modul netfilter i lodueshëm

lkm në hapësirën e kernelit të Android. Një *lkm* është një filë në gjuhën C ku thjesht 2 funksione kryesore janë të dallueshme: *init_module()* për të iniciuar modulin kur ai është i instaluar në pajisje dhe modulin *cleanup module()* për ta hequr atë nëse duam ta heqim nga funksionimi apo të sjellim përseri një modul më të optimizuar.

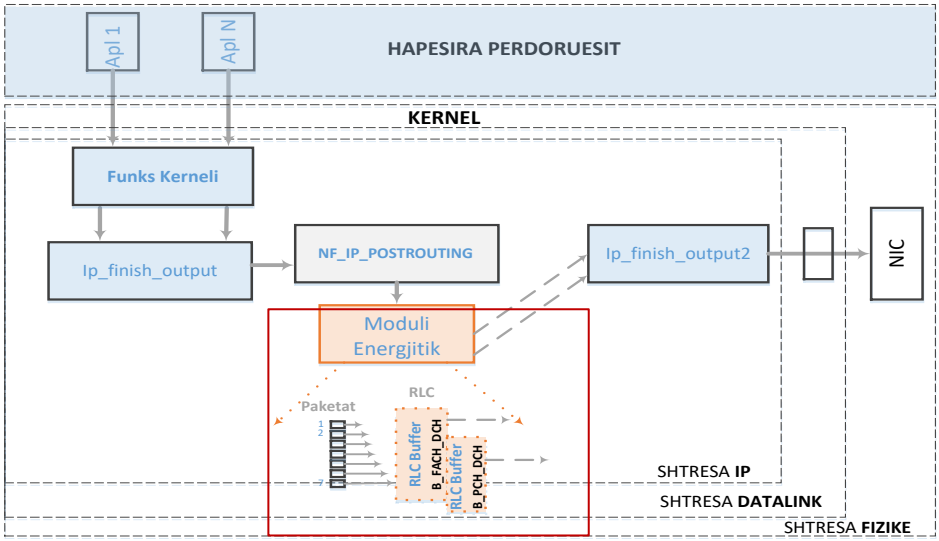


Figura 7.13: Arkitektura për vonesën dhe planifikimin e paketave në shtresën e rrjetit

Figura më sipër tregon implementimin e modulit brenda shtresës së rrjetit të Linux. Blloqet mësipërme në figurë kanë funksione kernel në shtresën IP. Në anën tjetër, blloku NF_IP_POSTROUTING i referohet grepave (hooks) netfilter të përcaktuar në protokollin IPv4. Për më shumë shigjetat tregojnë drejtimet që një paketë mund të ndjekë gjatë kalimeve të saj. Në fund pjesa portokalli tregon modulin e ri i cili është një patch file i ri ose i quajtur ndryshe i ndërmjetëm. Siç mund të vëzhgojmë, moduli është vendosur pas grepit NF_IP_POST_ROUTING dhe përpara funksionit *ip_finish_output2()*, që nënkupton që paketat dalëse do të kalojnë përmes modulit tonë përpara së të lënë pajisjen. Kështu që ne mund të përcaktojmë të zgjedhim trafikun dalës sikurse tregohet dhe në pjesën e poshtme të figurës me procesin e analizimit të 7 paketave.

Moduli në mënyrë të thjeshtë konsiston në tre ndarje: *kryesore*, *riimplementim* dhe *kontrolli i gjendjes UE*. Për më shumë informacion, shihni figurën 7.14 si më poshtë.

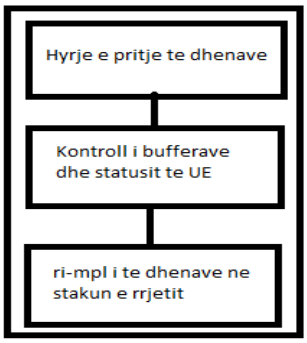


Figura 7.14: Struktura e përgjithshme e zgjidhjes

Seksioni kryesor përmban grepin netfilter si dhe një mbajtës të pritjeve, i cili performon apo ka funksionin për dedektimin dhe pritjen e paketave dalëse. Një grep netfilter është i rregjistruar duke përdorur një strukturë nfho të tipit nf_hookfn. Moduli ne fakt regjistron një grep NF_IP_POST_ROUTING për paketat IPv4, e cila është grepi i fundit përpara se paketat të lënë pajisjen. Në këtë lloj mënyre, ne kapim të gjitha paketat e daljes pasi kalojnë në fazën e rrugëzimit, në mënyrë që ato të kenë adresat e tyre destinacion dhe të gjithë grupin e parametrave të rrjetit. Çdo grep ka një funksion grepi i cili është kërkuar çdo herë që një paketë dalëse arrin grepin. Ky funksion vendos fazën tjetër të kapërcimit të paketës. Në rastin tonë paketa mund të pranohet ose vendoset në pritje. Vlerësuesi i pritjes: Ai është komponenti apo pjesa në funksion që menaxhon rradhën e paketave. Kështu, ne duhet të përcaktojmë 2 rradhë të ndryshme për pritjen e paketave *të vogla* - V dhe *të mëdha - M*. Paketat janë vënë në pritje në 2 lista të lidhura apo ndryshe memorje te brendshme (P_M dhe P_V) në varësi të madhësisë së tyre. Seksioni rinjektimit apo i rifutjes së paketave përsëri në transmetim nga memorizimi, është përgjegjës për transmetimin e paketave të vëna në pritje/memorje dhe updatimin e tranzitimit të gjendjeve së UE bazuar në volumin e të dhënave të transmetuara dhe është në gjendje fjetjeje kur pritjet apo listat janë bosh. Seksioni i kontrollit të gjendjes së UE tranziton kur një vlerë kohore e pasivitetit të UE mbaron pa dedektimin e ndonjë transmetimi. Në veçanti ky funksion përditëson (update) tranzitimin e gjendjeve të UE të FACH ose DCH, në varësi të volumit të të dhënave çdo herë që një paketë transmetohet. Kur një paketë arrin grepin dhe UE nuk është në kanalin DCH, paketa është vënë në rradhë duke thërritur funksionin enqueue() etj. Ky funksion kërkon madhësinë e paketës dhe e vë në rradhë atë në rradhitjen korresponduese. Për më shumë funksioni i kontrollit të gjendjes së UE mat se sa shumë kohë ka kaluar nga transmetimi i fundit dhe kryen update-in e gjendjes së UE kur kohëzuesit e joaktivitetit T_1 ose T_2 mbarojnë pa kryer transmetimin e ndonjë pakete. Në lidhje me vendosjen e tyre në pritje dhe memorizimin e përkohshëm në buffera, përdoren ato që quhen listat e lidhura në linux. Linux-i implementon një standard listash të lidhjeve dyfishe që ka një tërësi avantazhesh.

7.4 Implementimi në pajisjen UE bazë Android

Së pari pajisja UE duhet të jetë pa limitime (pra rooted) sepse ne duhet të ndryshojmë parametrat default të kernelit dhe duam privilegje si “superuser - su”. Përsa UE është *rooted (të drejta su në linux)*, zhvillimi i zgjidhjes sonë konsiston në 6 hapa kryesorë të cilat janë standarte në kësi rastesh:

- **a.** Mbledhja e të dhënave SW OS Android të Linuxit përkatës.
- **b.** Shkarkimi i kodit burim të kernelit në pajisjen UE dhe ndër-kompilimi me toolsin ARM.
- **c.** Konfigurimi i kernelit me opsionet e kërkuara nga imazhi i ri i kernelit dhe ndër-kompilimi i kernelit për të mbajtur imazhin e ri të kernel.
- **d.** Ngarkimi (boot-imi) i kernelit të ri në UE me adb apo toolse specifik si Odin.
- **e.** Ndërkompilimi i modulit tonë me kernelin dhe ngarkimi në UE.
- **f.** Kontollojmë me aplikime OS logs të UE sonë për ndonjë gabim/error në sistem.
- **g.** Instalime aplikimesh Android të nevojshme për testime.
- **h.** Heqja e modulit nësë duhet gjatë apo në fund të procesit.

Megjithatë mund të ketë devijime nga ky proces apo dhe metoda të tjera në varesi të vendorëve të prodhuesve smartphone. E rëndësishme është që moduli i ri të shikohet si pjesë e strukturës apo

funksioneve të Netfilter në punën e UE gjatë transferimeve data. Një ndihmesë të madhe në këtë proces kanë dhënë dhe informacionet e marra nga forumi XDA në web.

7.4.1 Hapat e përgjithshëm të portimit të UE

Pajisja UE Android që do të jetë nën testim duhet të jetë root-ed, me tolse specifike si tools-in në PC për UE Android *Revolutionary* apo ClockworkMod + SuperSU i cili kyc në OFF sigurinë e ruajtjes të brendshme te NAND, për të lejuar processin e "flashing" të imazheve. Për pajisjet smartphone Samsung ClockworkMod jep mundësinë e krijimit të backup dhe restore të imazheve SW [135]. Duhet kontrolluar që toolsi ADB punon në rregull dhe të kemi mundësuar më parë opsionin "USB debugging" në UE poshtë. Gjithashtu duhet të shkarkojmë një aplikim Android *Superuser* në kartë memorjeje të jashtme *sdcard* për të marrë të drejta të plota në UE, pra të na ofrohet mundësia e komandave të brendshme $su e cila në Linux-in që ka Android nuk është e mundur. Duhet të jemi me të drejta të plota sepse shumë aplikime në Android kërkojnë të drejta SU për tu egzekutuar.

Ne gjithashtu duhet të ri-startojmë UE në "recovery mode" i cili është specifik për smartphone/ modele të ndryshem. Imazhi SW Android i UE (apo kerneli bazë i njohur ndryshe si zImage) duhet të jetë downloaduar me parë në web-et zyrtare të prodhuesve HW [126]. Konfigurimet e kernelit të imazhit të UE duhet të modifikohen nga opsionet default për të lejuar ngarkimin e imazhit të ri. Për këtë një tools për ndër-kompilime duhet për të ndërtuar përsëri kodin burim të kernelit së bashku me imazhin e ri. Ky tools është i mundur të merret dhe të jetë në përputhje me SW Android [145].

Opsione të ndryshme të kernel duhen ndryshuar dhe kjo nëpërmjet komandave duke qënë në direktorinë bazë të kernelit. Konfigurimet e kernel sikurse kemi thënë ruhen në filen *.config*. Ne duhet të lejojmë opsionin e lejimit të ngarkimit të modulit të ri si dhe të lejojmë ndryshimet në netfilter (nëpërmjet editimit ti files *.config*). Me anë të toolsit të ndërkompilimit më pas ne ndërkompilojmë kernelin burim me komandën editor të UE. Ky proces duhet të përfundojë me sukses dhe ku imazhi i ri i kernelit është krijuar në direktorin *arch/arm/boot/zImage*. 1234 jep vitin e përshtatshëm për ndërkompiluesin.

$ make ARCH= arm CROSS_COMPILE=arm -1234 q1/bin/arm -none - linux - gnueabi -

Që moduli ynë i shkruar në gjuhën C të instalohet në UE ai duhet të ndërkompilohet. Pikërisht për këtë ne duhet të krijojmë një file tjetër Makefile, dhe ky file së bashku me modulin tonë duhet të jenë në të njëjtën direktori për të dhënë komandën për ndërkompilimin e modulit dhe përfitimin e tij në format linux (*.ko*).

Pasi imazhi i ri i kernelit dhe moduli ynë janë gati, pajisja UE duhet të ri-startohet në opsionin *bootloader* dhe më pas të mundësohet ngarkimi / bootimi nga kerneli i ri. Pas kësaj mund të vendosim modulin tonë në direktorin *data* në telefon. Pas kësaj aktivizimi i modulit duhet të kryhet me anë të komandës *insmod*

 # adb reboot bootloader
 # fastboot boot zImage
 # adb push poweralb.ko /data/ poweralb.ko

 # adb shell
 # insmod /data /poweralb.ko
 # rmmod /data /poweralb.ko

Megjithatë në më shumë detaje do të flitet më poshtë në punim, ku specifikisht përdoret dhe toolsi Odin për të ngarkuar imazhin e ri apo toolsi CWM (clockworkmod).

7.4.2 Ndërtimi i librarisë libnetfilter_queue

Eshtë një bibliotekë userspace që siguron një API për paketat që kanë qenë në radhë apo pritje nga filtri i paketës së kernel. Ky modul i gatshem merret në internet. *Libnetfilter_queue* varet në *libnfnetlink*, kështu që duhet të shkarkojmë të dyja libraritë tek http://www.netfilter.org/projects/libnfnetlink/downloads.html. Pasi t'i kemi shkarkuar të dyja, i ekstraktojmë të dyja libraritë në projektin/dosjen e OS Android në dosjen *.jni*. Kopjojmë filën *nfqnl_test.c* nga folder *libnetfilter_queue-1.0.0/utils/* te folderi .jni. Dhe krijojmë një file *Andoid.mk* me përmbajtjen si më poshtë (procedurë e njëjtë si për modulin e ri *.co* te seksioni 7.6.1):

```
#LOCAL_PATH is used to locate source files in the development tree.
#the macro my-dir provided by the build system, indicates the path of the current directory
LOCAL_PATH:=$(call my-dir)

#             build libnflink                     #
================================================================
include $(CLEAR_VARS)
LOCAL_MODULE:=nflink
LOCAL_C_INCLUDES := $(LOCAL_PATH)/libnfnetlink-1.0.0/include
LOCAL_SRC_FILES:=\
    libnfnetlink-1.0.0/src/iftable.c \
    libnfnetlink-1.0.0/src/rtnl.c \
    libnfnetlink-1.0.0/src/libnfnetlink.c
include $(BUILD_STATIC_LIBRARY)
#include $(BUILD_SHARED_LIBRARY)

#             build libnetfilter_queue             #
================================================================
include $(CLEAR_VARS)
LOCAL_C_INCLUDES := $(LOCAL_PATH)/libnfnetlink-1.0.0/include \
    $(LOCAL_PATH)/libnetfilter_queue-1.0.0/include
LOCAL_MODULE:=netfilter_queue
LOCAL_SRC_FILES:=libnetfilter_queue-1.0.0/src/libnetfilter_queue.c
LOCAL_STATIC_LIBRARIES:=libnflink
include $(BUILD STATIC LIBRARY)
#include $(BUILD_SHARED_LIBRARY)

#             build our code                      #
================================================================
include $(CLEAR_VARS)
LOCAL_C_INCLUDES := $(LOCAL_PATH)/libnfnetlink-1.0.0/include \
    $(LOCAL_PATH)/libnetfilter_queue-1.0.0/include
LOCAL_MODULE:=nfqnltest
LOCAL_SRC_FILES:=nfqnl_test.c
LOCAL STATIC LIBRARIES:=libnetfilter queue
LOCAL LDLIBS:=-llog -lm
#include $(BUILD_SHARED_LIBRARY)
```

```
include $(BUILD_EXECUTABLE)
```

Më pas japim cmnd "ndk-build" për të ndërtuar libraritë dhe ekzekutuar filën e ekzekutueshme *nfqnltest.* Por gjithsesi libnetfilter është pjesë e Android dhe merret e mirëqënë në funksionalitetin e tij.

7.4.3 Ekzekutimi i kodit ./nfqnltest në Android.

Një rast egzekutimi i një kodi i cili bën que / pritje të paketave. Për egzekutimin japim komandat në terminalin linux si më poshtë, ku kemi komanda për të kopjuar dhe ekzekutuar në pajisjen Android dhe e egzekutojmë atë si për shembull:

- *$ adb shell*
- *$ su*
- *$ mkdir /data/data/nfqnltest*
- *$ chmod 777 /data/data/nfqnltest*
- Hap një terminal tjetër. Shko te libs/<armeabi*> folder në projektin Android.
- Vendos komandën: "*adb push nfqnltest /data/data/nfqnltest/"*
- Kalo te terminali i parë dhe vendos komandën :*"cd /data/data/nfqnltest"*
- *./nfqnltest*

Për të konfiguruar rregullat e iptables, mund të hapim një terminal të ri dhe të ndjekim komandat si më poshtë:

 # adb shell
 # su
 # iptables –A OUTPUT –p tcp –j NFQUEUE –queue-num 0

Dhe në terminalin që bëjmë run *nfqnltest,* mund të shohim outputet e programit. Nëse hapim një browser aplikimi në pajisjen smart UE dhe provojmë psh. Google.com, do të shohim disa informacione paketash të shfaqura:

 # pkt received
…….
 # hw_protocol=0x0000 hook=3 id=0 outdev=12 payload_len=288
 # entering callback
 # pkt received
 # hw_protocol=0x0000 hook=3 id=1 outdev=12 payload_len=869
 # entering callback

Me *libnetfilter_queue*, mund të bëjmë shumë gjëra në interes, si psh NAT-ing, përgjim dhe kapje të paketave (packet sniffing & capturing) që është në interesin tonë etj.

Pasi pajisja UE është ngarkuar me SW e ri te modifikuar të lejojë fila të reja në kernel dhe me modulin e inkorporuar në të, duhet të merremi me testimet për të nxjerrë në dukje përfitimet energjitike dhe problematikat që mund të sjelli moduli i ri. Duhet thënë se kur ngarkohet SW i ri me modulin e ri, pajisja inteligjente UE duhet të ngarkohet nëpërmjet memorjes së jashtme /sdcard dhe për aplikimet që ne duam të testojmë si Facebook, Droidwall, Tshark etj.

7.5 Hapat e ndërmarrë për fillimin e modulit

Eshtë shumë i rëndësishëm procesi i inicimit të modulit. Në momentin që moduli është ngarkuar në pajisjen UE me bazë Android, funksioni kryesor inicion/starton dhe regjistron grepin Netfiltër si dhe mbajtësin e pritjes, të cilët së bashku formojnë hapin e dedektimit dhe vënies në pritje të paketave. Për këtë arsye, funksionet e rifutjes dhe kontrolli i gjendjes së UE janë filluar/nisur. Figura 7.15 riparaqet hapat e inicimit të modulit *poweralb.ko*.

```
#############################################################################################
1: Fillim
2:  Ngarkim i modulit
3:  UE ne PCH
4:  rregjistrim Grepi
5:  {
6-1:# dedektim i paketave hyrese
6-2:    # procesit te ri-futjes
6-3:       # i kontrollit te gjendjes se UE
7   }
8: Heqje e modulit
9: Fund
#############################################################################################
```

Figura 7.15: Hapat e ndërmarrë për inicimin e modulit

Ne mund të vëzhgojmë se dedektimi i paketave ardhëse, mbajtësit të rifutjes dhe kontrollit të gjendjeve të UE, janë në punë në harmoni dhe punojnë paralelisht. Hapi i dedektimit të paketës është përgjegjës për interceptimin dhe radhitjen e paketave që mbërrijnë në grep. Figura 7.16 përshkruan funksionalitetin në këtë hap.

```
#####################################6-1#########################################################
1:  Dedektim i paketave hyrëse
2:  statusi i UE
3:  {
4:    UE ne DCH ?
5:# Po → transmeto paketën
6:# Jo → mat madhësinë e paketës
7:          {
8:           Paketa > Bufferi B2 (FACH - > DCH ) ?
9:               #Po → vendos në pritje PM
10:              #Jo → vendos në pritje PV
11:         }
12:              # kontroll i statusit të ri-injektimit
13:  ri-injektimi i gjumit = "i vertete"
14:                  #Jo → shko te hapi 1
15:                  #Po →vendos rifutjen e zgjimit = "i vertete"
16:  Thirr nje thread ri-futje
17:    thirr funksionin në vazhdim (seksion 3)
#############################################################################################
```

Figura 7.16: Dedektimi paketash

Në kohën që paketa mbërrin te grepi, gjendja e UE kontrollohet. Nëse gjendja e UE është në kanalin *DCH* paketa pranohet, ndryshe paketa vihet në rradhë. Pastaj, madhësia e paketës kontrollohet. Nëse madhësia e paketës është më e madhe se B_2, paketa është vënë në rradhë në P_M, përndryshe në P_V. Për këtë arsye, është kontrolluar nëse funksioni *reinject* është në gjendje të fjetur

ose jo. Nëse funksioni riimplemetimit është në gjumë dhe aty ka paketa në rradhë, variabli riimpl i zgjimit vendoset në statusin “i vërtetë” dhe funksioni riimpl zgjohet.

Më poshtë jepet realizimi në filën apo modulin përkatës funksioni për kontrollin e gjendjes së UE dhe kushtin për vendosjen e tyre në memorizim apo pritje. Kontrollohet nëse në kanalin DCH ku të gjitha pritjet do të transmetohen, në të kundërt vendosi në pritje.

```
                                                    /* funksion te therritet nga Grepi */
unsigned int hook_func(unsigned int hooknum, struct sk_buff *skb, const struct net_device *in,
                       const struct net_device *out, int (*okfn)(struct sk_buff *))
{
    struct timespec tspec;
    struct iphdr *network_header;             /* deklarim i leximit te paketave te header-it */
    network_header = (struct iphdr *)skb_network_header(skb);
    if((network_header->protocol==1)||(strcmp(skb->dev->name,"lo")==0))
    {
        return NF_DROP;
    }
    else
    {
        if(uestate_current==DCH)                    /* Kontroll nese UE eshte ne kanalin DCH */
        {
            getnstimeofday(&tspec);
            write_lock_bh(&queue_lock);
            gtlast=tspec.tv_sec;
            write_unlock_bh(&queue_lock);
            return NF_ACCEPT;
        }
        else
        {
            return NF_QUEUE;
        }
    }
}


static void enqueue_packet(struct nf_queue_entry *entry)
{
    if((entry->skb->len)>=B_FACH_DCH)             /* nese madhesia e paketes e me madhe se kufiri B_FACH_DCH */
    {                                             /* paketa vihet ne rradhe ne Ql */
        list_add_tail(&entry->list, &ql);
        sdeadline=sdeadline_Ql;
    }
    else                                          /* ne te kundert vihet ne Qs */
    {
        list_add_tail(&entry->list, &qs);
        sdeadline=sdeadline_Qs;
    }
}

```

Figura 7.17: Kontroll i gjendjes/kanalit së UE vënia në pritje

Më pas vjen rradha e funksionit të rifutjes së paketave të kapura dhe futjes së tyre në transmetim përsëri. Duhet thënë se funksioni riimplementimit është fillimisht në gjendje përgjumjeje apo jovepruese. Më pas e ndron statusin nga 0 në 1 duke e nxjerrë nga statusi në përgjumje në aktive. Dhe më pas duke kërkuar për kohën dhe për kë ka lidhje apo ka mbaruar ky afat limit (për paketat e mëdha apo të vogla).

Figurat mëposhtë janë nxjerrë prej modulit në C dhe japin një pjesë për realizimin e këtyre funksioneve:

```
113  static int enqueue(struct nf_queue_entry *entry,unsigned int queuenum)
114  {
115
116      enqueue_packet(entry);                    /* vendos paketat ne pritje ne  */
117      if(reinject_sleep)                        /* zgjon mbajtesin reinject ne gjume */
118      {
119          write_lock_bh(&queue_lock);
120          reinject_wakeup=1;
121          reinject_sleep=false;
122          wake_up_interruptible(&reinject_wqh);
123          write_unlock_bh(&queue_lock);
124      }
125      return 1;
126  }
127
128
129  static void reinject_queue(int queue_list)
130  {
131      struct nf_queue_entry *entry, *next;
132      struct timespec tspec;
133      unsigned int npackets=1;
134      if(queue_list==Ql)
135      {
136          list_for_each_entry_safe(entry, next, &ql, list)
137          {
138              list_del(&entry->list);
139              if(entry && entry->skb)
140              {
141                  nf_reinject(entry, NF_ACCEPT);
142                  getnstimeofday(&tspec);
143                  write_lock_bh(&queue_lock);
144                  gtlast=tspec.tv_sec;
145                  write_unlock_bh(&queue_lock);
146              }
147          }
148      }
149      else if(queue_list==Qs)
150      {
151          list_for_each_entry_safe(entry, next, &qs, list)
152          {
153              list_del(&entry->list);
154              if(entry && entry->skb)
155              {
156                  nf_reinject(entry, NF_ACCEPT);
157                  getnstimeofday(&tspec);
158                  write_lock_bh(&queue_lock);
159                  gtlast=tspec.tv_sec;
160                  write_unlock_bh(&queue_lock);
161              }
162          }
163      }
164  }
```

Figura 7.18: Zgjimi i funksionit të rifutjes ne transmetim

Derisa funksioni rifutjes është zgjuar, transmetimi i paketave fillon. Kur transmetimi i paketave ka mbaruar funksioni i rifutjes në stakun e rrjetit është vënë përsëri në gjumë. Figura 7.19 tregon grafikun e fluksit që korrenspodon me hapin e transmetimit bazuar në gjendjet e UE të modelit të planifikimit, shpjeguar më parë:

```
##########################################################################################################
a:#Thirrur nje thread ri-injektimi
b: #Prit
1:  >Start
2:   {
3:     ri-injektimi i zgjimit = "i vertete" ?
4:      #Po →vendos "ri-injiketim i gjumit" = Fals
5:        {
6:           Ka paketa në PM & PV ?
7:              #Jo → kontrollo statusin e UE
8:                 {
9:                    UE = DCH ?
```

```
10:                     #Po → tranmeto  PM & Pv
11:                         UE = DCH
12:                         Shko to hapi 1
13:                     #Jo → Kontrollo Statusin e UE (shko te 6)
13:                         UE = FACH ?
14:                           #Po → shuma = 0 dhe kontrollo
15:                           Pv = 0 byte ?
16:                               # Jo →Shko te hapi 1
17:                               # Po → Transmeto paketën e 1 ne Pv
18                                   shuma n = shuma n-1 + "madh Paketës" dhe kontrollo
19:                                     shuma n > B2 (FACH → DCH) ?
20:                                       #Jo →transmeto paketën dhe
21:                                             Shko te hapi 1
22:                                       #Po → Prit kohe vonese Kp
23:                                             Shko te hapi 1
24:                                #Jo →në cilën Px ka mbërritur paketa
25:                                  Prit kohen Kafert
26:                                  Mbërritur në Pm ?
27:                                   #Po →Transmeto PM dhe co
28:                                          UE = DCH &
29:                                          Shko to hapi 1
30:                                   #Jo →llogarit shumën në PM
31:                                          SM > B1 (PCH → DCH) ?
32:                                                #Po → shko hapi 27
33:                                                #Jo → llogarit shuma totale në Pv
34:                                                    Sv > kufiri limit ?
35:                                                      #Po → shko hapi 27
36:                                                      #Jo → transmeto Pm
37:                                                         UE = FACH, prit Kp
38:                                                       Shko te hapi 1
39:                 }
40:   #Po →"ri-injektusi zgjimit" = fals
41:                  "ri-injektues i gjumit" = true
42:                    #Prit  (hapi b)
43:                      #shko te hapi  1
44:       }
45:   #Jo → thirr hapin 41
46:  }
47: fund
##########################################################################################
```

Figura 7.19: Proceset për rifutjen e paketave

Njëkohësisht me funksionin e rifutjes dhe grepin Netfilter është duke punuar edhe funksioni i kontrollit të gjendjes së UE. Hapat e mëposhtëm riparaqesin fluksin e kontrollin të këtij mbajtësi brenda modulit të ri. Ai ruan kohën kur paketa e fundit është transmetuar dhe periodikisht performon tre kërkime baze:

Nëse UE =DCH dhe koha e bishtit $T_T > T_1+T_2$ e ndodhur për cdo transmetim: gjendja e UE kyçet në PCH (pasi një komunikim i ri ndodhet jashtë kohës së mundshme të transmetimit apo brenda Tt) kur kushtet përmbushen dhe kërkohet periodikisht statusi i UE. Në të kundërt, thërret hapin tjetër si: Nëse UE=DCH dhe koha bishtit $T_T > T_1$ and $T_T < T_2$: gjendja e UE kyçet në FACH kur kushtet përmbushen dhe kërkohet periodikisht statusi i UE (pra kur një komunikim i ri bie brenda kohës së gjëndjes së komunikimit në PCH). Në të kundërt, thërret hapin tjetër:

```
int uestate_control_thread(void *unused1)                /* kontroll i kanalit te UE */
{
    struct timespec tspec;
    while(1)
        {
        msleep(3);
        uestate_last=uestate_current;

        getnstimeofday(&tspec);
        gtnow=tspec.tv_sec;

        if((uestate_last==DCH)&&((gtnow-gtlast)>=(T1+T2)))          /* nese UE ka qene ne DCH dhe ka kaluar koha T1+T2  */
        {
            write_lock_bh(&queue_lock);
            uestate_current=PCH;                                    /* UE eshte ne PCH */
            write_unlock_bh(&queue_lock);
        }
        else if((uestate_last==DCH)&&((gtnow-gtlast)>=T1)&&((gtnow-gtlast)<=(T1+T2)))   /* nese ka qene ne DCH dhe koha me madhe se T1 e me e vogel se T1+T2 */
        {
            write_lock_bh(&queue_lock);
            uestate_current=FACH;                                   /* UE eshte ne FACH */
            gtlast=gtnow;
            write_unlock_bh(&queue_lock);
        }
        else if((uestate_last==FACH)&&((gtnow-gtlast)>=T2))        /* nese UE ka qene ne FACH dhe ka kaluar koha T2 */
        {
            write_lock_bh(&queue_lock);
            uestate_current=PCH;                                    /* UE eshte ne PCH */
            write_unlock_bh(&queue_lock);
        }


    }
        return 0;
}

static void init_uestate(unsigned int state)
{
    if(state==DCH)
    {
        uestate_current=DCH;
        uestate_last=uestate_current;
    }
    else if(state==FACH)
    {
        uestate_current=FACH;
        uestate_last=uestate_current;
    }
    else if(state==PCH)
    {
        uestate_current=PCH;
        uestate_last=uestate_current;
    }
```

Figura 7.20: Kontrolli i gjendjes së UE dhe kohëve në komunikim

Nëse UE = FACH and $T_T > T_2$: gjendja e UE kyçet në PCH (pasi komunikimi ka mbaruar tail-et apo kohëzuesit e pasivitetit dhe detyrimisht shkon në PCH) kur kushtet përmbushen dhe kërkohet periodikisht statusi i UE. Nëse jo, kërkohet periodikisht statusi i UE.

Moduli në netfilter jepet në shtojcën B3 ku një grup parametrash duhet të vendosen paraprakisht në modul përpara se të kompilohet. Në raste ndryshimesh të parametrave moduli duhet të rikompilohet përsëri dhe thjesht të futet në stakun e OS Android me komandën *insmod*. Nëse pas matjeve specifike nuk realizohet një ruajtje e energjisë së kërkuar nga limitimi i transmetimeve në shtresën e rrjetit moduli mund të hiqet nga kerneli pra dhe nga funksionet.

7.6 Vlerësime dhe rezultatet

Qëllimi kryesor i këtij seksioni është të përcaktojë që moduli ynë operon në rregull dhe ruan energji. Megjithatë testohen opsionet për paketa UDP të vogla, të mëdha dhe aplikimin Facebook. Para çdo eksperimenti, pajisjet do të jenë të përgatitur për të përmbushur kushtet e mëposhtme:

- Bateria e UE do të jetë plotësisht e ngarkuar ~100 %;
- Përdoruesi i UE të jetë në pozicion statik, pra jo në lëvizje.
- Karta SIM do të jetë me opsionin 3G. Lejojmë vetëm ndërfaqen e rrjetit 3G/4G me Data ON.

- Pajisja është ri-startuar. WiFi, Bluetooth dhe GPS do të jenë të gjitha të çaktivizuar.
- Të gjitha proceset opsionale në sfond do të jenë ndërprerë / përfunduar. Vetëm programet e nevojshme për sistemin operativ do të jenë funksionale dhe mbeten aktive; perdoret TaskManager si dhe DroidWall për të eleminuar ndonjë ndërveprim ekstra të padëshiruar.
- Gjithashtu ekrani të jetë i fikur gjatë testimeve, për të eleminuar konsumin nga ekrani. Në rastet e testimeve me paketa UDP mund të nevojitet që ekrani dhe touch të jenë on.
- Sinkronizimi i aplikimeve është kthyer "off". DroidWall të konfigurohet për aplikimet e duhura.
- GSam api do të përdoret dhe për të mbledhur të dhëna të konsumit të energjisë.
- Të dhëna të tjera në lidhje me fuqinë apo kohëzuesit në kanal janë marrë nga toolse apo api me telefona të tjerë.

Kjo bëhet e gjitha për të minimizuar riskun për ndërhyrje të jashtme të cilat mund të ulin saktësinë e rezultateve të testeve.

7.6.1 Pajisjet Vlerësuese dhe Programet

Në mënyrë që të performojmë vlerësimin e testeve, ne duhet të përdorim burime të ndryshme që janë shpjeguar në tabelën 7.3, 7.4 dhe 7.5.

Pajisje	Përshkrimi
HP Elitebook 2540p (Ubuntu) HP EliteBook (Windows 7.0)	Ne kemi përdorur këta 2 modele kompjuterash PC / laptop në Ubuntu14.04 LTS dhe Windows 7.0 për të zhvilluar dhe matur karakteristikat e zgjidhjes tonë.
Smasung S3/4 GT-I9100 (Android OS me Linux Kernel 2.6.37)	Smartphone Android modern me aplikime ndihmëse matëse profesionale si Nemo Handy, Netmonitor etj
Nokia Lumia 625 (Windows 8.1 OS)	Smartphone është përdorur për matjet e energjisë direkte në telefon për kryerjen e testeve të ndryshëm.
+355 66 6xxxx SIM card	Kartë SIM nga operatorët mobile ne Shqiperi për matjet e 3G.
Skema me rezistorin 0.22 Ohm	Ne e kemi përdorur këtë për të amplifikuar rënien e tensionit brenda qarkut që lejon matjet e konsumit të energjisë në UE. Matet gjithashtu dhe vlera aktuale ne batteri me nje
Digital meter GW Instek – GDM-8246 sampling board	(0.02% DCV saktësi) për të matur tensionin nëpër baterin e telefonit dhe gjithashtu rënien e tensionit nëpër rezistencën e vendosur e cila është e lidhur me pajisjet nëpërmjet një kablli 2-fijësh sikurse në figurën 5.23
Digital Multimeter DT9205A	Pajisje matëse e rënies së tensionit në një rezistencë të lidhur paralel me baterinë e telefonit.

Tabela 7.3: Lista e pajisjeve HW të nevojshme për testime

Një tërësi aplikimesh Android [79], Nokia e gjithashtu dhe për PC duhen për të na ndihmuar për të kryer matje specifike dhe për më shumë:

Aplikimet	Përshkrimi
SuperSU	Utility për të drejat su- të pajisjes Andorid.
ClocworkMod	Utility për recovery te pajisjes Andorid.
Terminal Emulator	Aplikim Android bazë Linux për të egzekutura komanda në Linux.
SysLog	API që logon aktivitetet e OS të Android.
UDP client (sender)	UDP client është një aplikim Android (në internet) që gjeneron një trafik UDP Uplink.

iPerf	Aplikim Androdi UDP që mundëson të dërgojmë ose marrim datagrama UDP te klientë të tjerë që përdorin të njëjtin aplikim, ose lejon dëgjimin e paketave UDP. Mesazhet mund të dërgohen si ASCII ose Hex.
UDP TCP Server	Aplikim Android për të dërguar komanda UDP/TCP nga pajisja testuese në një pajisje tjetër UDP/TCP të mundshme. Për ndërfaqet WiFI dhe 3G.
DroidWall	DroidWall është një aplikim firewall Android që mundëson lejimin-bllokimin e ekzekutimit të aplikimeve në telefon.
bitShark	Shark është një gjurmues / sniffer Android që kap paketat e rrjetit në telefon.
tPacketCapture	Bën kapjen e paketave dhe pa pasur telefonin me të drejta të plota përdorimi (root permission)
NEP, Nokia Energy Profiler	Nokia tools për matjen e energjisë dhe kohëzuesve në telefon. Mat fuqinë, kohëzuesit dhe timerat.
NetworkLog	Aplikim Android per te mbledhur te dhëna per paketat e shkëmbyera (dërguara e marra) si dhe shpejtesite ne rrjet
PowerTutor	Aplikim Android i përdorur për të kryer matje të energjisë për çdo algoritëm. Është një tools diagnostikues për analizim të sistemit dhe përdorim të fuqisë së aplikimeve. Aplikimi mund të ekzekutohet në çdo pajisje Android. Mund të ekzekutohet në background dhe ruan të dhëna / logs në lidhje me përdorimin e fuqisë për çdo aplikim. Informacioni i mbledhur është përmbledhur në një format përdoruesi të përdorshme. PowerTutor [7, 8] propozoi një metodë vlerësimi për komponentët hardware, por ai nuk ofron informacion energjie për çdo aplikim ose proces.
Current Widget	Aplikim Android për shfaqjen e fuqisë në UE.
OpenBattery	Ky aplikim mbledh imformacion rreth nivelit të baterisë së telefonit dhe sa shpesh e karikojmë atë. Të dhënat mund të dërgohen online në një server për këtë projekt. Ai mund të mbledhë të dhëna të nevojshme si lloji i telefonit, prodhuesi, koha e baterisë, temperatura, shkalla e ngarkesës, tensionit dhe kapacitetit etj. Këto informacione ruhen çdo herë që pajisja paraqet një ndryshim të gjendjes së baterisë, frekuenca e së cilës varet nga telefoni.
IP Tools: Network Utilities	Eshte nje Aplikim android për analizim dhe optimizim të rrjetave. IP e rrjetit merren prej këtij aplikimi.
Facebook	Aplikim përdoruesi SNS version
BBC News	Aplikim përdoruesi news
Gmail	Aplikim përdoruesi email
AccuWeather	Aplikim përdoruesi për motin
Gsam Battery Monitor	Aplikim për statistika të përdorimit të baterisë së UE

Tabela 7.4: Lista e aplikimeve të nevojshme për përdorim për testet (Google Play)

Në lidhje me 2 aplikimet e para, ne shpesh kemi nevojë për të kryer disa operacione dhe në pajisje na kërkohen të drejtat e përdoruesit root (kryesor). Në shumicën e imazheve për pajisjet Android komanda su- mungon dhe fjalëkalimi nga përdoruesi su- është i panjohur. Prandaj këto dhe disa aplikime të tjera ndihmojnë në eleminimin e këtij problemi.

Ekzistojnë dhe një sërë aplikimesh e tools-e HW dhe SW të cilat ndihmojnë në mbledhjen e të dhënave në lidhje me konsumin e energjisë si:

SystemSens [142]**:** Ky aplikim mbledh logs nga shumica e të dhënave në lidhje me një telefon dhe aplikimet e instaluara në të. Ajo mbledh logs për përdorimin e kujtesës, përdorimin e CPU, përdorimin e rrjetit dhe statistika të tjera të çdo aplikimi të ekzekutuar në telefon. Këto të dhëna transmetohen në mënyrë periodike në një server dhe mund të arrihen nëpërmjet një faqe interneti në server. Të dhënat shfaqen më pas në grafikë. Avantazhi i kësaj zgjidhjeje është se ajo ka gjithçka që nevojitet për të bërë një analizë të një aplikacioni. Ajo logon të dhënat në lidhje me një aplikim dhe gjithashtu i vizualizon këto të dhëna. Megjithatë, kodi është shumë i madh dhe i ndërlikuar. Serveri gjithashtu duhet të jetë i instaluar dhe kjo mund të rezultojë ndoshta në disa probleme dhe burime shtesë. Përveç kësaj, aplikimi nuk lejon zgjedhjen e frekuencave ku do të logojmë të dhënat. Një tjetër disavantazh është se ajo kap më shumë informacion se sa kërkohet për një studim.

Një listë programesh PC jepen si në tablën mëposhtë:

Programi	Përshkrimi
Eclipse	Eclipse është një mjedis/program zhvillues dhe programimi për disa gjuhë programimi si JAVA ose C ++. Përdoret për të integruar module të jashtme të përdorshme për testime apo matje të knosumit të energjisë në një UE .
Wireshark	Wireshark është një analizues paketash i rrjetit në kompjuter. Versioni i përdorur gjurmon paketat dhe për një telefon te lidhur me USB ne PC.
Android Studio, logcat	Android SDK është një tools/paketë e krijuar nga Google për zhvillimin e aplikimeve me bazë Android. Gjithashtu ky tools ka dhe një seksion të vecante për lidhjen e me UE si dhe simulatorë UEs të ndryshem.
ADB	Android Debug bridge si pjese e Android studio, SDK
Fastboot	Tool Android që bën flashing të SW dhe particione të tij, dhe vjen si pjesë e Android SDK.
Kies & Odin	Toolse PC për recovery të UE në raste të failure /crash SW.
Little Eye Labs	Aplikim PC e Android për të kryer matje të konsumit të energjisë në një pajisje UE Android të lidhur me këtë PC. Kërkon komunikim me SDK të Android për njohjen e UE nga USB.
Odin 1.85	Tools Gui për implementimin e firmwareve apo kerneleve te modifikuar ne UE.
NetperSec	Aplikim PC që monitoron shpejtësinë e të dhënave në PC për ndërfaqe LAN.

Tabela 7.5: Programet PC të përdorur

Android Software Development Kit, SDK [149] përmban një numër mjetesh për krijimin e aplikacioneve për Android OS. Këto mjete përfshijnë IDE (Android Eclipse-based Developer Tools dhe IntelliJ-based Android Studio), emulatorin (AVD - emulator Android Virtual Device bazuar në projekt QEMU), debugger-in (ADB - Android Debug Bridge), dhe toolse për montimin e aplikacioneve Java (Android SDK) dhe aplikacionet në C / C ++ (Android NDK). Aktualisht këto mjete janë në dispozicion për Linux, Mac OS X dhe platformat Windows.

Android Debug Bridge [148] është një tools në formatin komandë (cli) për t'u lidhur me pajisjen Android ose emulatorin që drejton sistemin Android duke përdorur kompjuterin të lidhur me pajisjen Android. Ai përbëhet nga tre komponentë: 1. klienti në makinën tuaj të zhvillimit (binar adb), 2. server, i cili shkon në sfond në makinë të zhvillimit, 3. daemon në pajisjen Android (ose emulator). Kur komanda *adb* thirret nga PC, së pari, do të kontrollojë nëse serveri në makinën e zhvillimit apo PC është në funksionim. Nëse jo, do të kryejë një procedurë "bootstrap" server dhe lidh atë në portin TCP. Kur serveri ngrihet dhe ekzekutohet,

ai dëgjon kërkesat për klientët në hyrje. Pas marrjes së një komande nga klienti, do të vendosë lidhjen me "daemon-in" e sfondit të pajisjes për të filluar komunikimin me pajisjen. Ka një sërë komandash për të komunkikuar, marrë dhe dhënë fila në PC.

LittleEyeLabs [144] është një tools profilizimi, që synon kryesisht në platformën Android. Në veçanti, ai ofron një funksionalitet për monitorimin e konsumit të energjisë të aplikacioneve të caktuara të instaluara në pajisjen Android. Përbëhet nga dy pjesë - aplikacioni i agjentit që duhet të instalohet në pajisjen Android dhe aplikacionin desktop që lidhet me agjentin në pajisjen Android dhe analizon rrjedhën e telemetrisë nga pajisja.

Sikurse dhe qëllimi i këtij punimi që të tregojë dhe vejë në dukje faktorët kryesorë në konsumin e shpejt të energjisë në smartphonët e sotëm, ne propozojmë e japim një zgjidhje në nivel të mesëm të OS Android (middleware - kernel) të transferimeve në sfond / background me vonesë dhe në nivel paketash në kushte apo kanale me konsum të ulët Energjie.

7.7 Testimet dhe Matjet reale

Para matjeve të energjisë në zgjidhjen middleware, ne kemi për të përcaktuar parametrat e nevojshme për sa i përket konsumit të energjisë të gjendjeve të ndryshme të UE, kohëzuesve të pasivitetit ose kufijtë limit të bufferave RLC. Këto vlera fillimisht duhet të jenë nxjerrë nga matjet të cilat përdoren si inpute për filën/modulin e ri për ruajtje të energjie nëpërmjet vonimit të transferimeve.

7.7.1 Vlera te parametrave radio nga matjet

Tabela e mëposhtme tregon adresat IP të përdorura në testime dhe të nxjerra prej aplikimeve wireshark në PC apo IP Tools në Android :

Pajisja	**Adresa IP**
UE Host	31.22.50.xxx
UE internal IP	10.248.169.126
Server testues ne Web	50.116.119.173
UE smartphone ()	31.22.50.xyz

Tabela 7.6: Adresat IP në testim (e ndryshueshme)

Tabela 7.6 dhe Tabela 7.7 tregojnë kufijtë limit të bufferave RLC (të matur me testim dërgim paketash në UL) dhe kohëzuesit e pasivitetit që përdoren për të kaluar/ndryshuar gjendjet UE për operatorin mobile, respektivisht:

Tranzitimi i gjendjeve	**Madhësia e UL (byte)**
Kalimi PCH-DCH	~ 530 +/- 20
Kalimi FACH-DCH	~ 270 +/- 20
Kalimi PCH-FACH	Trigeruar per transf te vogla

Tabela 7.7: Tranzitimi i gjendjeve dhe madhësitë e paketave për trigerim

Kohëzuesit e pasivitetit nga vëzhgimi me tool-set apo aplikimet e matjeve tregojnë se kalimet janë shumë të shpejta për këtë operator. Megjithatë shohim që pajisja shumë shpejt kalon në gjendjet Idle dhe vlerat për këtë operator janë:

Kohët e pasivitetit	Koha (sek)
T_1	2
T_2	5

Tabela 7.8: Vlerat e kohëzuesve të pasivitetit

Mesatarisht nga matjet e kryera në telefonin Smartphone Nokia E6 me NEP paraqesim një vlerë të matur të fuqisë mesatare radio për kanal komunikimi në ndërfaqen 3G, siç tregohet në tabelën 7.9 (duke konsideruar heqjen e fuqisë së kërkuar për Ekranin dhe CPUn ~ 400mW).

Parametri / kanal	Fuqia (mW)
Fuqia $_{IDLE}$	~ 10
Fuqia $_{FACH}$	~ 400
Fuqia $_{DCH}$	> 600
+ (Ekrani dhe CPU)	-

Tabela 7.9: Vlerat mesatare të konsumit të radio fuqisë (me PowerTutor)

Ne kemi marrë parasysh këto vlera për të kryer planifikimin e trafikut nëpërmjet modulit te ri. Në mënyrë të ngjashme, ne kemi përdorur edhe këto vlera në modulin tonë për të vlerësuar konsumin e energjisë e të traceve të matura.

Së fundi, moduli ynë përdor vlerën e dhënë Kv për transmetimin e paketave. Duhet thënë se, siç përmendëm dhe më parë një parametër ***Kv*** tregon minimumin e dritares së vonesës të dërgimit për transmetimin e paketave të vëna në pritje për tu transmetuar. Matjet e energjisë janë kryer për Kv = 60, 90 e 120 sek por sfidë do të ishte vëzhgimi i kursimeve të ndryshme të energjisë, kur Kv merr vlerë dhe më të gjatë. Ne e propozojmë një vlerë të Kv prej 60 sek bazuar dhe në studimin për periodicitetin e aplikimeve.

7.7.2 Parametra te nevojshëm te modulit

Tabela 7.10 tregon parametrat për modulin për operatorin celular nën testim. Operatori përdor kohë pasiviteti të vogla për protokollin RRC si dhe për pragjet e memorjeve RLC:

Parametri	Vlera
T_1 (s)	2
T_2 (s)	5
Memorjet RLC	**Vlera**
B_1 (byte) / P_M	530
B_2 (byte) / P_V	290
Kohet	**Vlera**
Kp (ms)	300
Kv *(s)*	60 & 120
Fuqite e kanalit	**Vlera**
P_{DCH} (W)	0.7
P_{FACH} (W)	0.4

Tabela 7.10: Parametrat e modulit të planifikimit të transmetimeve

Natyrisht këto parametra fillimisht duhet të përdoren në modulin e ri përpara se të implementohet në OS Android të smartphone-it nën testim. Sfidë në të ardhmen mbetet një update dinamik i disa prej tyre si p.sh ndryshimi i kohëzuesve të pasivitetit apo dhe kufijtë limit të tranzitimeve të cilat janë ekskluzivitet i operatorit mobile dhe mund të ndryshohen në rrjet. Ky update dinamik në modulin tonë mund të arrihet nëpërmjet po implementimeve të algoritmeve shtesë si pjesë e këtij moduli apo në moduli të dytë, për zbulimin e këtyre parametrave të rrjetit nëpërmjet kontrolleve periodike. Jo pak herë ndodh që këto parametra (kohëzuesat dhe memorjet) për arsye optimizimi nga ana e operatorit të ndryshohen dhe kjo do të sillte një devijim të funksioneve të modulit i cili i merr të paracaktara këto vlera të matura më parë në rrjet.

Në figurën poshtë po japim deklarimin e parametrave të përmendur në tabelën 7.9 në modulin per ruajtjen e energjisë:

```
// Modul SW Android per procesim me vonese te paketave ne 3G per eficence energjie

#include "/home/Lruci/smartphone/ossw/include/linux/kernel.h"          /* duhet per KERN_ALERT */
#include "/home/lruci/smartphone/ossw/include/linux/module.h"         /* duhet per gjithe modules */
#include "/home/lruci/smartphone/ossw/include/linux/moduleparam.h"
#include "/home/lruci/smartphone/ossw/include/linux/time.h"
#include "/home/lruci/smartphone/ossw/include/linux/types.h"
#include "/home/lruci/smartphone/ossw/include/linux/timer.h"
#include "/home/lruci/smartphone/ossw/include/linux/string.h"
#include "/home/lruci/smartphone/ossw/include/linux/netfilter.h"
#include "/home/lruci/smartphone/ossw/include/linux/netfilter_ipv4.h"
#include "/home/lruci/smartphone/ossw/include/linux/delay.h"
#include "/home/lruci/smartphone/ossw/include/linux/wait.h"
#include "/home/lruci/smartphone/ossw/include/linux/kthread.h"
#include "/home/lruci/smartphone/ossw/include/linux/spinlock.h"
#include "/home/lruci/smartphone/ossw/include/net/netfilter/nf_queue.h"
#include "/home/lruci/smartphone/ossw/include/net/ip.h"
#define DRIVER_AUTHOR "lruci Luan Ruci <luan.rucci@gmail.com>"
#define DRIVER_DESC   "Modul SW Android per procesim paketash ne 3G per eficence energjie"

#ifdef __KERNEL__
#ifdef CONFIG_NETFILTER

const unsigned int PCH=1;                /* kanalet 3G  */
const unsigned int FACH=2;
const unsigned int DCH=3;
const unsigned int Ql=1;                 /* Pritjet */
const unsigned int Qs=2;
const unsigned int Qs_FACH=3;
const unsigned int sdeadline_Ql=1;       /* afatet deadline te ruajtjes se paketave */
const unsigned int sdeadline_Qs=2;
static unsigned int sdeadline;
static unsigned int uestate_current;
static unsigned int uestate_last;
static struct nf_hook_ops nfho;          /* strukture qe mban nje grup te funksioneve te grepave */
static struct nf_queue_handler nfqh;
static DEFINE_RWLOCK(queue_lock);
static LIST_HEAD(ql);
static LIST_HEAD(qs);
static unsigned int ql_ptotal;
static unsigned int qs_ptotal;
static struct task_struct *reinject_task;
static struct task_struct *uestate_control_task;
static unsigned long int gtnow;
static unsigned long int gtlast;
const unsigned long int T1=1.5;            /* Kohezuesit e mos-aktivitetit (DCH-> FACH   e  FACH ne PCH ) */
const unsigned long int T2=5.5;
static unsigned int B_PCH_DCH=530;       /* Pragjet e RLC  */
static unsigned int B_FACH_DCH=294;
static unsigned int tclear=300;          /* ms, koha per fshirjen e bufferit RLC pas transm */
static unsigned int tw=60;               /* sekonda,  Kohe vonese e dergimite te paketave */

```

Figura 7.21: Parametrat hyrës të zgjidhjes së propozuar

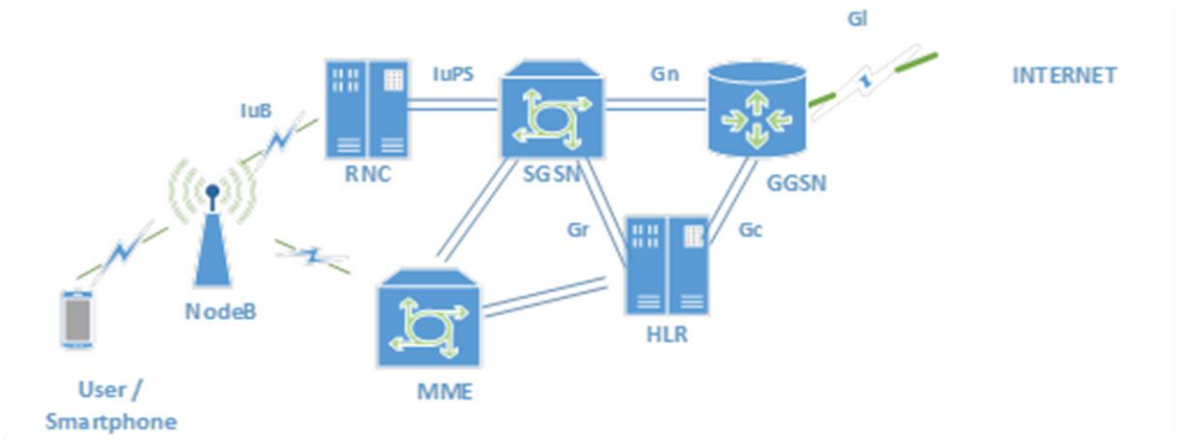
7.8 Matjet e ruajtjes së Energjisë

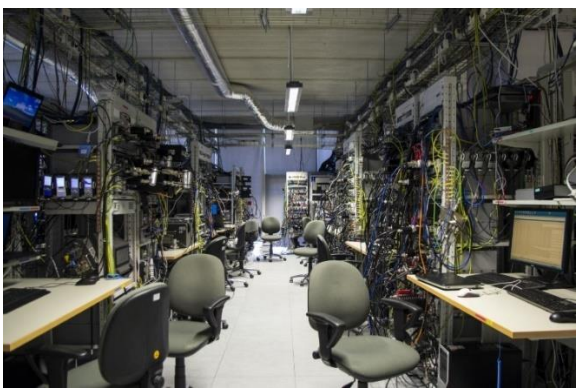
Ky seksion përshkruan matjet e konsumit të energjisë, për të përcaktuar kursimet e energjisë që zgjidhja jonë ofron. Ky test është i ndarë në dy pjesë kryesore, në varësi të madhësisë trafikut të rrjetit të përfshirë në matje dhe ne lidhje me aplikimet reale. Në pjesën e parë, matjet janë kryer me trafik të simuluar UDP për paketa të vogla e të mëdha, me dhe pa modulin. Prandaj, ne mund të bëjmë krahasime reale mbi këtë modul.

Në pjesën e dytë ne masim një trafik real të gjeneruar duke kapur paketat e shkëmbyera në një komunikim aplikimi real. Këto trace përmbajnë trafikun "best-effort" që nuk ka specifikime/ kërkesa të larta të QoS. Facebook është përdorur si aplikim i testimit dhe të gjitha tracet janë regjistruar me ekran të fikur dhe pa ndërveprimin e përdoruesit. Përderisa kursimi i energjisë të modulit varet nga sasia e trafikut, prej traceve të përdorura zgjedhim ato që janë për volume të ndryshëm trafiku dhe karakteristikash.

- Një seri testesh në Lab-in e Nokias në Poloni e Greqi (2014-2017).
- Një pjesë testesh në ambjente private në Shqiperi.
- Nje seri Toolsesh HW, SW e aplikime Nokia e Android.
- Telefona smart Nokia e Samsung të përdorur.

Matjet janë kryer duke qëndruar në pozicion fiks dhe jo të levizshem. Kështu që ndryshimet në sinjal apo qelizat e shërbimit nuk janë perfshirë ose kanë ndikuar në studimin tonë.





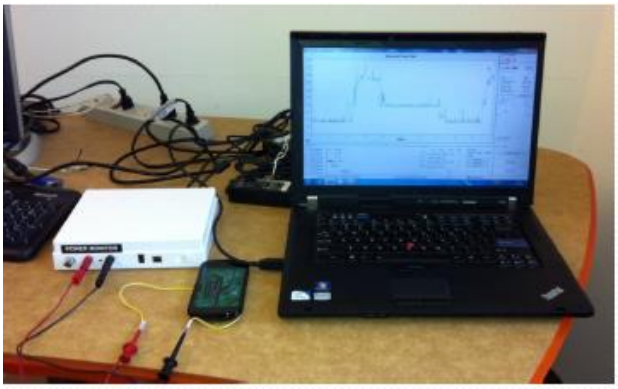


Figura 7.22: Infrastruktura e lab dhe matjet ne mjedise personale

7.8.1 Rezultatet e testimeve pa dhe me modulin ne Android

Në këtë seksion do të flasim për disa testime me një grup aplikimesh të lëna në background për një periudhë kohore, me ekran të fikur pra pa ndërveprimin e përdoruesit. Më pas të njëjtat teste do të bëhen me modulin e integruar në kernelin linux me një kohë pritjeje apo vonese prej 60, 90 e 120 sek. Pas kësaj do të bëhen krahasime për të nxjerrë në dukje përfitimin energjitik.

Një ide tjetër është parashikimi se si duhet të sillet moduli në lidhje me rastet e gjenerimit të trafikut të veçantë në uplink me kërkesën e klientit UDP. Rastet janë ndarë në dy raste në varësi të madhësisë së paketës së trafikut uplink: Vënie në pritje e paketave të mëdha dhe e paketave të vogla. Ne kemi vendosur një vlerë të vonesës Kv për 60, 90 e 120 sekonda për të kryer këto matje.

7.8.1.1 Matje ne Android Samsung *S3/4 pa zgjidhjen*

Në këtë rast testimi kryhet duke përdorur 1 dhe 4 aplikime Android dhe repsektivisht: Facebook, BBC News, Gmail dhe Acuweather. Aplikimet janë lënë aktive në background dhe të gjitha aplikimet e tjera janë mbyllur me TaskManager. Më pas në fund përdorim dhe GsamBattery për të përdorur disa vlerësime mbi konsumin për këtë periudhë kohore. Në fund kemi parë se cilat aplikime janë ende në background. Sic mund të shihet se Accuweather apo BBC News nuk i gjejmë në background pasi mbledhim të dhënat. Gjatë testit të natës kemi vënë re se në kohën e mbledhjes së të dhënave vetëm aplikimi Facebook qëndronte në background, dhe një pjesë e mirë e konsumit sikur në tabelën e 3 i dedikohet këtij aplikimi. Ndërkohë në tabelën e 4 kemi bërë një matje vetëm me aplikimin Facebook.

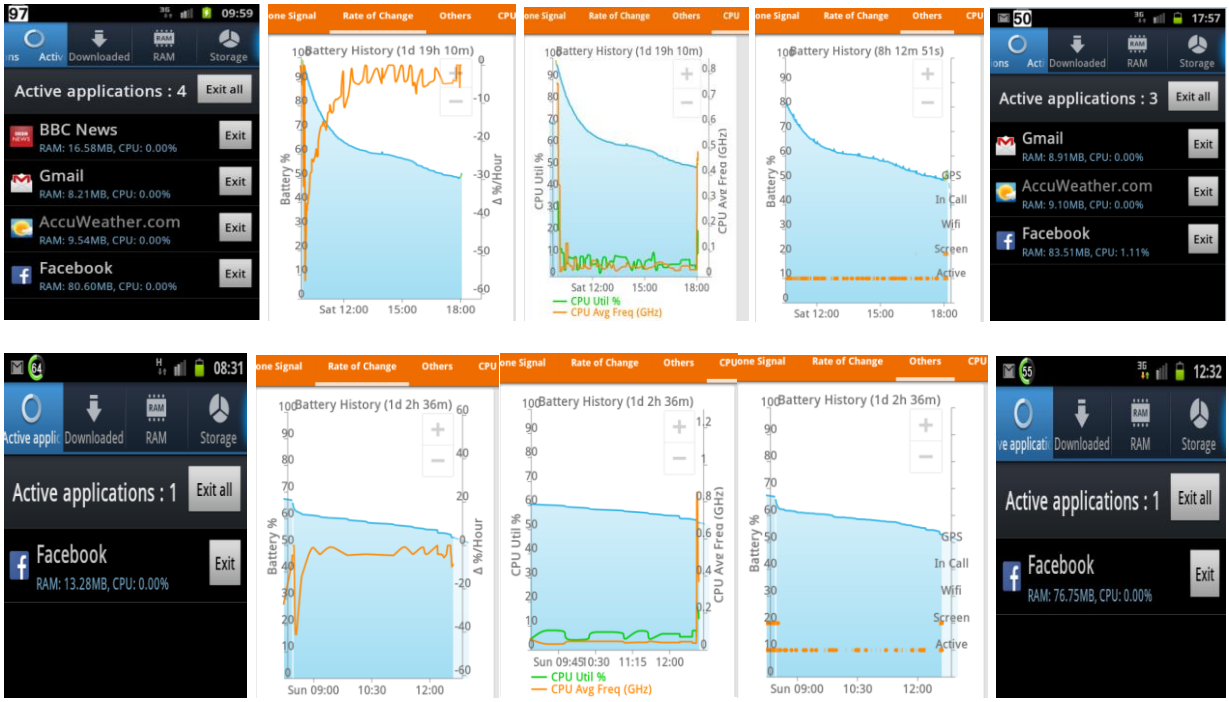


Figura 7.23: Të dhëna matjesh nga Samsung S3/4 I9100

Si rrjedhojë nga një analizim përmbledhës si në tabelën mëposhtë kemi :

Ora	*10^{00}*	18^{00}	(8 ore)
Niv Baterie	97 %	51%	-46 %
Data	(+android sistem)		**UL**
Aplikimet	*Facebook*		x
	Accuweather		x
	Gmail		x
	BBC News		x

Ora	*20^{00}*	22^{00}	(2 ore)
Niv Baterie	95 %	87%	-8 %
Data	(+Andr sistem)		**UL**
Aplikimet	*Facebook*		x
	Accuweather		x
	Gmail		x
	BBC News		

Ora	*23^{00}*	08^{00}	(11 ore)
Niv Baterie	95 %	72%	-23 %
Data	(+And Sistem)		
Aplikimet	*Facebook*		**x**
	Accuweather		
	Gmail		
	BBC News		

Ora	*08^{30}*	*12^{30}*	(4 ore)	
Niv Baterie	*64 %*	*55%*	-9 %	
Data	*Facebook + Andr Sistem*		UL - 137 KB	DL – 240 KB
Aplikimet				

Tabela 7.11: Konsumi energjisë për Api background pa zgjidhjen

7.8.1.2 Implementimi i zgjidhjes ne Android - Portimi

Sikurse kemi folur zgjidhja e propozuar implementohet si një modul kernel në format të përshtatshëm për kernel linux apo .ko i cili gjenerohet nga një file origjinale në *.C* apo *.C++* dhe në rastin tonë është fila *poweralb.c* i cili do të ndërkompilohet me toolsin ARM-2010 apo dhe 2012 sikurse më poshtë ku më parë duhet një Makefile i ri për filën tonë. Ky është një proces standart në rast se zgjidhja middleware apo kernel duhet të propozohen:

```
VERSION = 2

PATCHLEVEL = 6

SUBLEVEL = 35

EXTRAVERSION = .7-I9100XWKK5-CL754841

obj-m += poweralb.o

all:

make -C /home/luan/Desktop/sc02c_kernel-android-2.6.35.7-stable ARCH=arm CROSS_COMPILE=/home/luan/Desktop/arm-2010q1/bin/arm-none-linux-gnueabi- M=/home/luan/Desktop/TestModule/arm-2010q1/sc02 modules
```

```
clean:

make -C /home/luan/Desktop/sc02c_kernel-android-2.6.35.7-stable ARCH=arm CROSS_COMPILE=/home/luan/Desktop/arm-2010q1/bin/arm-none-linux-gnueabi- M=/home/luan/Desktop/TestModule/arm-2010q1/sc02 clean
```

Mësipër jepet direktoria ku ndodhet AOSP (apo sw bazë në Android) i Samsung S3/4 GT-9100, modeli i HW (ARM) si dhe direktoria ku ndodhet moduli ynë *poweralb.c*. Pasi egzekutohet ky Makefile në fund të procesit do to të gjenerohet një modul i përshtatshëm për kernel si poweralb.ko.

Por përpara se ky modul të implementohet si një modul i lodueshëm në Smartphone-in GT-I9100 , UE duhet të jetë i konfigururar në kernel (apo .config) për të lejuar ngarkimin e moduleve të rinj dhe specifikisht:

```
CONFIG_MODULES = y    (a)
```

Ky parametër ndodhet në filën .config e cila gjendet poshtë direktorisë root/include/. Ky file mund të thirret me adb dhe editohet por në rastet se kur kerneli është krijuar pa dhënë opsionin e editimit të .config atëherë një imazh i ri kernel duhet të krijohet me lejimin e tij si dhe opsioneve shtesë të kërkuara.

Meqënëse ne punojmë me një modul që ndërvepron në kernel dhe për më shumë në zonën e Netfiltër, duhet që gjithashtu dhe ky parameter të lejohet (enabled):

```
CONFIG_NETFILTER = y  (b)
```

Për të mundësuar këtë një seri hapash duhen kryer paraprakisht për të pasur të përshtatshëm apo modifikuar kernelin e Android të GT-9100. Specifikisht duhet të punohet mbi AOSP (source code-in e GT-9100 kernel ver 2.6.35.7) Për të mundësuar ndryshimin e adb mësipër. Për këtë në PCn me Ubuntu 14.04 LTS dhe Android SDK, egzekutojmë një grup komandash që na mundësojnë modifikimet e nevojshme si dhe krijimin e një kerneli të ri i cili mund të implementohet në Smartphone GT-I9100 [126] me toolsin ODIN sikur në hapat mëposhtë:

```
$ sudo su :   (password)

# cd /GT-9100 dir/

# ls -lrt

# chmod 777 -R *.*

# make clean && make mrproper

# make  ARCH=arm  c1_rev02_defconfig

# make menuconfig  (GUI Kernel  apo .conf dhe modifikojme settings per a) dhe b) )

Ver 1 (script):

# ./build_kernel.sh

  (ne perfeundim te procesit poshte direktorise /arch/arm/boot/  do te gjendet kernel I ri zImage I modifikuar/pershattur per ndryshimet qe ne duam)

Ver 2 (cmdn make):
```

```
# make -C /home/luan/Desktop/sc02c_kernel-android-2.6.35.7-stable ARCH=arm CROSS_COMPILE=/home/luan/Desktop/arm-2010q1/bin/arm-none-linux-gnueabi-

# tar -xvf new_kernel.tar.gz zImage
```

Tashmë imazhin e ri të kernel zImage duhet ta ngarkojmë në Smartphone-in tonë më anë të toolsit Odin. Paraprakisht UE duhet të fiket dhe të ndizet në Odin Mode (Vol buton Down + Main + Power && Vol UP). Më pas në toolsin Odin do të shfaqet i lidhur dhe në seksionin PDA vendosim Kernelin e ri new_kenrel.tar dhe e iniciojmë UE me kernelin e ri. Procesi si në figurë:

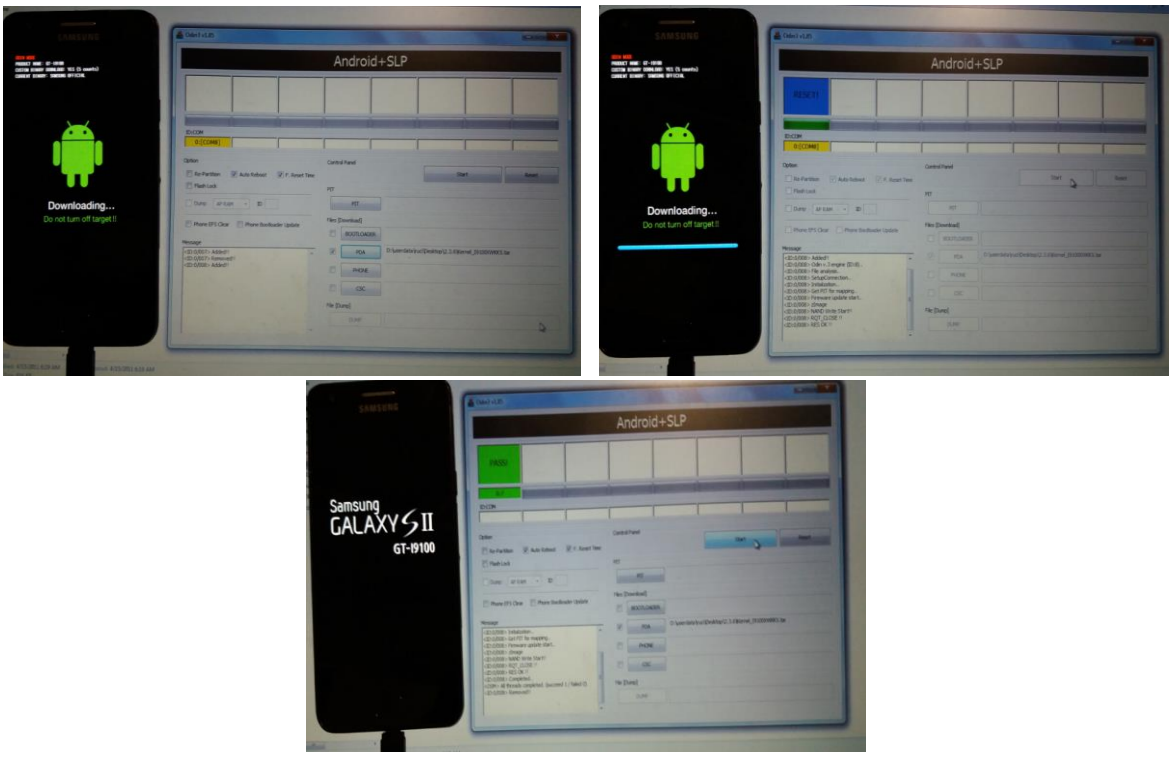


Figura 7.24: Ngarkimi i kernelit të modifikuar në UE S3/4 GT-I9100 [105]

Pasi kerneli i ri është ngarkuar duhet të përdorim toolsin ADB te Android SDK (në nën-direktorin e SDK të platform-tools/adb) i cili mundëson lexim, ngarkim e shkarkim filash në kernelin e Smartphone-it tonë. Modulin tonë e kopjojmë në direktorin ku kemi toolin adb. Në këtë mënyrë mund të fusim apo heqim modulin në UE:

```
# cd /android sdk adb tool/

# adb devices

# adb pull .....

# adb push poweralb.ko   /root/temp

#adb shell

$ cd /root/temp

$chmod 777 poweralb.ko

$ insmod poweralb.ko
```

Nese duam ta heqim

$ rmmod poweralb.ko

Shikohet nga afër nëse nuk ka error gjatë ngarkimit të modulit. Gjithashtu përsëri UE duhet të root-ohet duke përdorur aplikimin CWM dhe Superuser si dhe më pas të instalohen të gjithë aplikimet e tjera që kërkojnë SU dhe do të përdoren për matje apo monitorim si tshark apo tpacketcapture.

7.8.1.3 Matje ne Android Samasung S3/4 me zgjidhjen

Në këtë seri matjesh do të përsëriten të njëjtat teste sikur më parë me aplikim/aplikime në background pasi moduli specific për planifikim transmetimesh të dhënash në UL të jetë implementuar në kernelin linux të OS Android. Merren në konsideratë tre raste me vonesë 60 e 90, e 120 sekonda.

Aplikimi GSam është përdorur për të mbledhur të dhëna dhe për ti krahasuar ato me rastin pa zgjidhjen e implementuar. Shikohet një reduktim i energjisë së kërkuar, porse nga 4 Api në background vetëm 2 prej tyre u gjendën përsëri në background sikur në tabelë. Nga vëzhgimi me GSam u pa se një pjesë e mirë e konsumit përsëri i dedikohet NET IF dhe aktiviteteve të kernel. Krahasuar me tabelën e matjeve pa implementimin e zgjidhjes vërehet një reduktim i energjisë, ndoshta jo shumë domethënës por që tregon se ka mundësi për përmirësime energjitike në UE.

Ora	*23^{00}*	09^{30}	(11 ore)
Niv Baterie	95 % 99 %	72% 79%	23 %\ 20 %\
Data	(ne bckg)		**UL**
Aplikimet	*Facebook* *Accuweather* *Gmail* *BBC News*		x x

Aplikimet	**Kvon**	**Kohe ne background**	**Konsumi energjise**
Facebook, Gmail, Weather, BBC News	-	10 h	99% -> 78% (- 21%)
Facebook, Gmail, Weather, BBC News	60 sek	10 h	99% -> 87% (- 12%)
Facebook, Gmail, Weather, BBC News	-	4 h	99% -> 90% (-9%)
Facebook, Gmail, Weather, BBC News	60 sek	4 h	99% -> 94% (-5%)

Aplikimet	Kvon	Kohe ne background	Konsum energjise
Facebook, Gmail, Weather, BBC News	-	10 h	99% -> 78% (~21%)

Facebook, Gmail, Weather, BBC News	90 sek	10 h	99% -> 91% (~8%)
Facebook, Gmail, Weather, BBC News	-	4 h	99% -> 91% (-9%)
Facebook, Gmail, Weather, BBC News	90 sek	4 h	98% -> 94% (-4%)

Aplikimet	Kvon	Kohe ne background	Konsumi energjise
Facebook, Gmail, Weather, BBC News	-	10 h	99% -> 78% (~21%)
Facebook, Gmail, Weather, BBC News	120 sek	10 h	99% -> 92% (~7%)
Facebook, Gmail, Weather, BBC News	-	4 h	99% -> 91% (-9%)
Facebook, Gmail, Weather, BBC News	120 sek	4 h	98% -> 95% (-3%)

Tabela 7.12: Konsumi energjisë për Api background me zgjidhjen Kv=60 e 120s

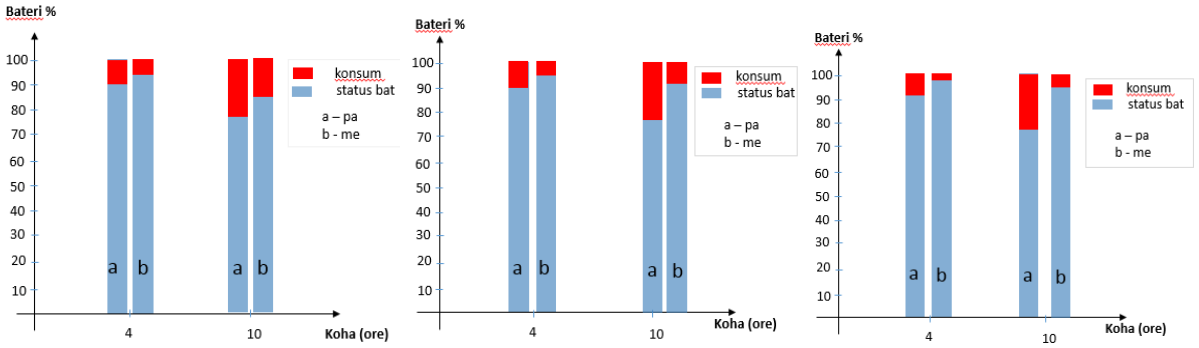


Figura 7.25: Konsumi energjisë për Api background me zgjidhjen Kv =60, 90 &120s

7.8.1.4 Parashikimi nga vënia në pritje e paketave të mëdha

Qëllimi i vënies në pritje të paketave të mëdha është të tregojë se UE kyçet në gjendje nga PCH për te DCH, kur vëllimi i të dhënave tejkalon pragun kufi të memorjes RLC FACH - DCH (Tabela 7.9). Simulimi do të supozojë në dërgimin e një pakete prej 570 byte çdo 1.5 sekonda. Figura 7.26 tregon pritshmëritë një krahasim mes mënyrës me dhe pa zgjidhjen (zgjedhjen e transferimeve në uplink).

Ne mendojmë se për testin me planifikim e vonim transferimesh, moduli bën “queue” dhe transmeton të gjitha paketat në pritje pas 60 e 120 sek të Kv, dhe gjendja e UE kalon direkt te DCH për shkak të madhësisë së paketës që e kalon pragun FACH - DCH të bufferit RLC. Përveç kësaj, gjendja e UE do të mbetet më pak kohë në DCH dhe jo shpesh, që do të thotë më pak konsum të energjisë.

Aplikimet me protokoll UDP janë më fleksibel për zgjidhjen ndërsa ato me TCP kërkojnë një kthim përgjigje dhe vonesa e krijuar shkakton disa shqetësime në këtë lloj komunikimesh me këtë protokoll. Kjo llogjike përkon me llogjikën e komunkimimeve në 4G po ashtu.

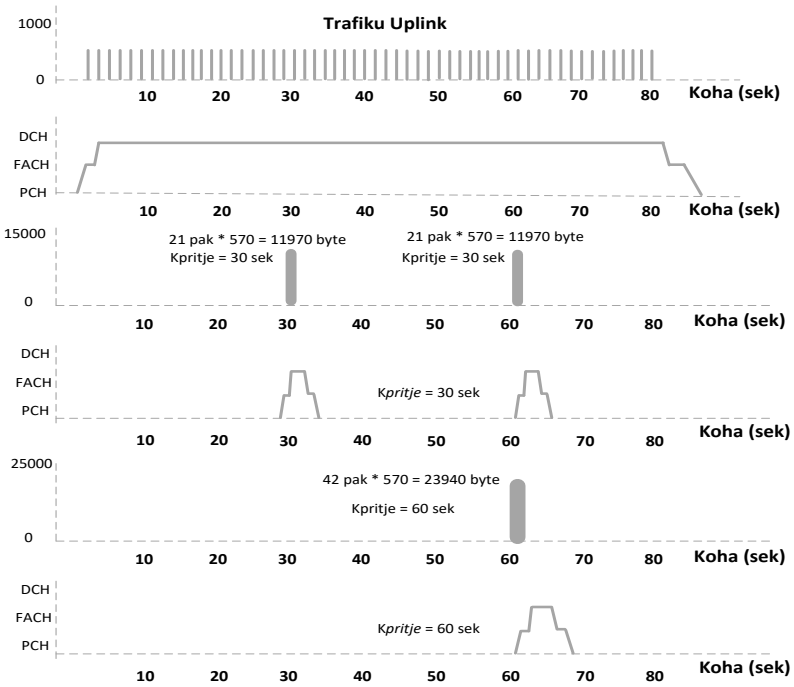


Figura 7.26: Simulim i vënies në priteje i paketave të mëdha

7.8.1.5 Simulim nga vënia në pritje e paketave të vogla

Në të kundërtën, nëse vemë në pritje paketa të vogla për të verifikuar se UE do të ndryshojë gjendje nga PCH në FACH, kur vëllimi i të dhënave nuk kalon kufirin FACH - DCH të bufferit RLC (tabela 7.8).

Testi është i njëjtë si më sipër, përveçse madhësia e paketave të transmetuara është 63 bytes. Figura 7.26 tregon një krahasim mes simulimit origjinal uplink dhe atij te pritshëm të planifikuar nga moduli.

Ne mendojmë se testi me planifikim e vonim nga moduli, vë në pritje dhe transmeton të gjitha paketat me një 60 sek të Kv, dhe UE kyçet në FACH në vend të DCH për shkak të madhësisë së paketave që nuk e kalojnë pragun/kufirin FACH - DCH të bufferit RLC.

Për më tepër, ne gjithashtu vërejmë se gjendja UE duhet të mbetet më pak në kohë në FACH në ne rastet me zgjidhjen se sa në rastin pa zgjidhjen.

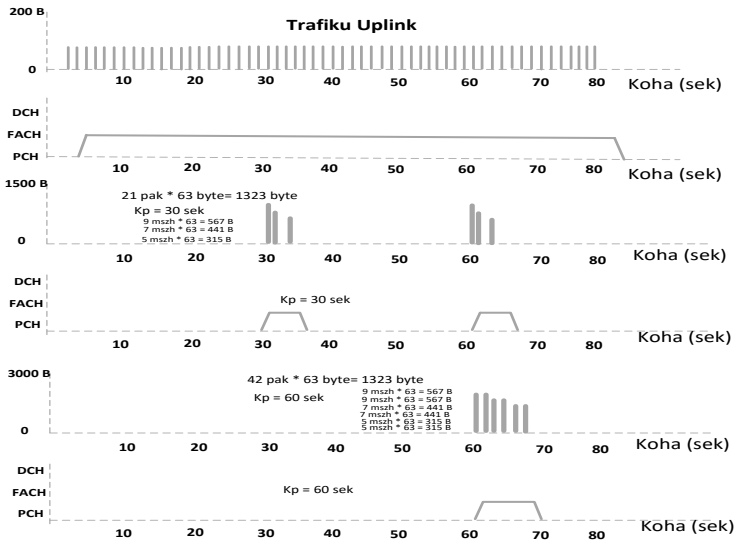


Figure 7.27: Simulim i vënies në pritje të paketave të vogla

Nëse dërgojme një trafik cdo 5 sek (brenda T1 & T2) me paketa me madhësi si në tabelë. Të gjitha aplikimet janë të bllokuara përveç API për dërgim paketash.

Trafik	Madh pakete	Kanali aktiv	Kvon	Ruajtje E (%)
Trafik UL	>B2 = ~290 B	FACH	60 sek	~ 10 -> 30 %
Trafik UL	>B2 = ~290 B	FACH	120 sek	
Trafik UL	< B2 = ~290 B	DCH	60 sek	NA
Trafik UL	< B2 = ~290 B	DCH	120 sek	
Trafik UL	< B2 = ~290 B	FACH	60 sek	~ 20 -> 45 %
Trafik UL	< B2 = ~290 B	FACH	120 sek	

Tabela 7.13: Permbledhje nga simulimi i trafikut te te dhenave

Në përfundim, mund të themi se operacioni i vendosjes në pritje të paketave me anë të modulit nuk mund të konsumojë ndonjë fuqi të ndjeshme dhe prandaj kemi një përfitim të energjisë.

7.8.2 Rezultatet përmbledhëse të ruajtjes së energjisë

Ne kemi llogaritur kursimet e energjisë në lidhje me kushtet bazë (dmth, kur kemi e nuk kemi transferim me vonesa në burst) me ndryshime në afatin e paketave. Ne kemi vendosur një kohë vonese dhe Kv te 60,90 e 120 sek kua pas asaj të gjithë transmetimet janë si një burst i vetëm.

Rezultatet e matjeve tregojnë që koha që UE qëndron në FACH apo DCH për kanale me konsum më të lartë të energjisë, është më e shkurtër në rastin e transmetimeve me planifikim. Në këtë lloj mënyre reduktohet energjia e shpenzuar. Tabelat mëposhtë tregojnë konsumin e energjisë dhe ruajtjen e energjisë për matjet e trafikut të simuluar dhe real. Duhet te theksojmë që energjia e

konsumuar në cdo trace është e matur me dhe pa modulin. Kur moduli është i egzekutuar përdoret një vlerë e Kv prej 60, 90 e 120 sek.

Tip Trafiku	API's background	Kohe Matjesh bckg	Konsumi E
Normal	4 Api	10 h	~21%
Me filen / von= 60 sek	4 Api	10 h	~12 %
Me filen von= 90 sek	4 Api	10 h	~8 %
Me filen von = 120 sek	4 Api	10 h	~5 %

Tabela 7.14: Konsumi i fuqisë i matur për trafikun me dhe pa modulin sw

Matjet janë nxjerrë nga API Gsam Battery, i cili mundëson mbledhje të dhënash dhe i shfaq ato sipas kushteve të ndryshme:

- pasi UE të jetë shkëputur nga USB.

Por duhet të përmendim se nga vëzhgimi ynë:

1. Rezultatet e matjeve janë të variueshme nga settings të updateve në cdo API.
2. Ngarkesa e rrjetit apo shpejtësitë për kanal ndikojnë.
3. Konsumit i dedikohet Net IF & aktivitetit Kernel (CPU etc)

Sikurse dhe në shumë punime në këtë fushë për algoritme të tjera, tipi i trafikut në rastin pa planifikim të transmetimeve konsumon një energji të konsiderueshme e cila s'duhet neglizhuar. Nga analiza e matjeve duket qartë se trafiku me planifikim konsumon më pak energji sesa rasti pa modul. Duhet theksuar se besojmë se sa më gjatë të vendoset vlera e kohës se vonimit Kv në sekonda aq më shumë ruhet energjia, por kjo sjell problematika të QoS (vonesa, ritransmetime, dublikime ACK etj.)

Përfitimi E	**~ 5 - 20% (për trafik normal në bckg)** **~ 10 – 40 % (simulim të trafikut)**

Tabela 7.15: Perqindje të ruajtjes së energjisë në rastin me modulin aktiv

Mund të shprehim se energjia e konsumuar nga moduli është shumë më e vogël në krahasim me shumën totale të energjisë të konsumuar në rastin e transmetimit pa modul, dhe për këtë ai mund të konsiderohet i pandjeshëm në shumën totale të energjisë.

Shuma e energjisë bazë të ruajtur nga moduli do të ishte pas një zbritjeje të shumës totale të energjisë së ruajtur me energjinë e konsumuar vetëm nga përdorimi i modulit. Për shkak të konsumit minimal sikurse thamë energjia e ruajtur nga planfikimi i transmetimit të trafikut është shumë afër energjisë së ruajtur të rrjetit (apo totale) si shkak i energjisë shumë të ulët që konsumon moduli.

KAPITULLI VIII

8.1 Konkluzione

Pajisjet smartphone kanë evoluar me shpejtësi të lartë duke sjellë shumë funksionalitete e aplikime të cilat në anen tjetër kanë rritur presionin mbi energjinë e kërkuar në UE. Teknologjitë e rrjetave të operatoreve mobile në anën tjetër kanë një ndikim mbi konsumin e shpejtë dhe për shkak të parametrizimeve në komunikim. Por ndërkohë në anën tjetër nga analizimi i trafikut 3G e 4G, një pjesë e madhe e trafikut në sfond është krijuar nga aplikimet për shkak të mos qenit "të vetëdijshëm" për karakteristikat energjetike të komunikimit. Kjo si rrjedhojë e problemeve në programimin e aplikimeve pa këndvështrimin energjitik e të mbingarkesave në rrjet e sinjalizim, por dhe shumë aplikimeve të tjera që kanë në bazën e vet proceset për shkëmbimin periodik të mesazheve të mbijetesës duke krijuar një ndryshim të shpeshtë gjendjes së ndërfaqes së rrjetit.

Në këtë punim ne kemi kryer matje fizike si dhe me anë te aplikimeve në dy telefona smartphone Nokia e Samsung, dhe kemi evidentuar fushat e konsumit të shpejtë të energjisë dhe kemi faktuar si të rëndësishme izolimin e konsumit të shpejtë të energjisë të transferimeve të të dhënave që varen nga faktorë të tjerë që lidhen me parametra të rrjetit apo mënyrën e punës së protokolleve të rrjetit. Në veçanti, ne tregojmë se kohëzuesit e pasivitetit të gjendjeve të protokollit RRC, kufijtë e të dhënave të buferave në RLC dhe mbulimi radio i rrjetit në mënyrë direkte influencojnë konsumin e energjisë së transfertave të të dhënave 3G e 4G.

Të dhënat e bufferave RLC në uplink dhe downlink janë përdorur për të llogaritur vëllimin e trafikut për të shkaktuar trigerimet e tranzicionit të gjendjeve apo kanaleve më të larta energjitike në ndërfaqen Uu. Kur të dhënat në buffer tejkalojnë pragun, RNC rivlerëson shpërndarjen e burimeve dhe kërkon ndryshim në gjendjen e UE sikurse kemi përmendur. Ne i kemi matur pragjet/kufijtë uplink kryesisht që shkaktojnë tranzicionet e treguara në këtë punim, duke testuar duke dërguar paketa UDP të madhësive të ndryshme dhe duke vëzhguar energjinë faktike aktuale të kërkuar për transmetime në këto kanale radio. Eshtë vërejtur se profili energjitik i UE në një komunikim na mundëson të kuptojmë se do të jetë më mirë që për transferime të vogla të përdorim kanale me energji më të ulët si ato FACH pa qënë nevoja të tranzitojmë direkt në kanale me energji të lartë si DCH.

Ne studiuam më në detaje arkitekturën e protokolleve të ndërfaqeve të rrjetit si Uu, IUB, IUPS, S1U e S1-MME dhe analizuam funksionet e protokolleve RRC e RLC për proceset e promovimit dhe demontimit të gjendjeve gjatë dhe pas një transferimi të dhënash. Eshtë parë se aktualisht protokolli RRC si për të dhëna me kapacitet të ulët si dhe për të dhëna me kapacitet të lartë mund të tranzitojë direkt në kanale DCH me energji të lartë, si dhe po ashtu dhe kohët e kthimit mbrapsht kanë një ndikim negativ në performancën energjitike të smartphonëve. Nga matjet kanali Cell_FACH është një konsumues më i ulët sesa kanali Cell_DCH por me shpejtësi më të ulët, prandaj një planifikim eficent i transmetimeve në FACH e DCH pas një vonese do të sjellë një përfitim energjitik. Gjithashtu do të shmangen dhe tranzitimet e shpeshta midis kanaleve duke ulur dhe sinjalizimin në rrjetin e operatorit.

Propozohet një zgjidhje për planifikimin e transmetimit të të dhënave kryesisht në sfond për drejtimin uplink si dhe me nje vonesë te tyre (60, 90 e 120 sek). Planifikimi i transmetimeve në një smartphone kërkon dhe një njohuri më të detajuar të komunikimit të shtresës radio midis UE dhe rrjetit të operatorit. Nje algoritëm që vendos në pritje paketat dhe i transmeton ato me kushte në lidhje me zgjedhjen e kanaleve dhe kjo për të përmirësuar kursimin e energjisë në krahasim me punën pa zgjidhjen e propozuar. Paraprakisht për algoritmin duhet të jenë matur disa parametra të rrjetit për komunikimin UL dhe ato duhet të përdoren si informacion hyrës për funksionin e algritmit i cili është implementuar si një zgjidhje e mesme SW në një model smartphone Android.

Për këtë një njohje e mënyrës se ku duhet implementuar propozimi në Android OS ka qënë një prej sfidave që në kemi hasur. Nga studimi kemi parë se struktura Netfilter brenda kenrelit Android ofronte opsionet e nevojshme për kapje, analizim e përpunim paketash. Prandaj implementimi në brendësi të struktures netfilter si një file e vetme apo e shpërndarë si disa shtesa apo modifikime në shumë funksione të filave egzistuese të netfiltër ishin opsionet e para.

Ne e implementuam si një file të re duke i përmbledhur të gjitha funksionet në një filë të vetme në mënyrë që dhe në të ardhmen modifikimet e duhura ti kryejmë mbi këtë file pa ndërhyre për ndryshime në shumë fila të tjera. Kjo filë e programuar në gjuhën C lindi nevojën për kompilim dhe portimin e saj si një modul sw kernel netfiltër në një smartphone me bazë Android. Një sërë toolsesh e aplikimesh Android si për kompilim, portim e matje janë përgatitur e sistemuar bashkë në një PC me bazë linux për të na ardhur në ndihmë me implementimin dhe matjet më pas. Procesi i kompilimit nga një file .c në një filë .ko mundëson krijimin e një moduli energjitik si një zgjidhje e mesme SW në Android.

Moduli sw kernel prodhon një vonesë në komunikim, sepse paketat janë vënë në pritje përpara se të transmetohen. Kjo vonesë mund të ndikojë në punën e disa aplikimeve dhe protokolleve të rrjetit që janë të ndjeshëm nga vonesat. Një seri matjesh janë kryer për të parë diferencën me dhe pa zgjidhjen e propozuar.

Matjet tona janë kryer në intervale të ndryshëm e në periudha të ndryshme prej vitit 2014 - 2017. Ato janë kryer në një pozicion fiks dhe në distancë nga antenat shërbyese me një sinjal jo shumë të fortë (~ 95dbm) si dhe për një operator të ngarkuar ku shpejtësia e të dhënave në DCH (~ disa qindra kbps) nuk është shumë e madhe për shkak të ngarkesës që ka rrjeti. Eshtë vendosur një vonesë në transferime burst prej 60, 90 e 120 sekondash dhe kjo ka treguar se sa më lartë vonesa aq më i madh është përfitimi energjitik pasi smartphone mbetet më gjatë në Idle apo në FACH. Efektiviteti është më i lartë për aplikime background.

Nga matjet është vërejtur se sa më të mëdha të jenë afatet limit të vonesave, aq më shumë mund të arrihen kursimet e energjisë në transmetim, por kjo padyshim nga vëzhgimi krijon disa ritransmetime të paketave apo dhe dublikime. Zgjidhja e propozuar është eficente për transferime të dhënash në background apo sfond dhe që nuk kanë volume të larta trafiku, si dhe nuk implikon ndryshime në aplikimet egzistuese në smartphone. Nga matjet rezulton një ruajte e energjisë që shkon nga 15 deri në 45 %.

Megjithatë ky punim tregon se zgjidhjet SW të mesme janë një avantazh dhe zgjidhje të shpejta kur kërkohet një perfitim në një fushë apo drejtim të caktuar. Mjafton të dihet ku dhe cfarë duhet

ndërhyhet e të përmiresohet. Në rastin tonë ishte përfitimi energjitik nga një reduktim i tranzitimeve në kanale energjitik të lartë e me vonesa dhe implementimi si një zgjidhje patch kernel.

8.2 Puna në të ardhmen

Moduli merr prioritet në rastet kur përdoruesi është jo aktiv apo pasiv, dhe nuk ndërvepron me telefonin. Por se shumë aplikime janë në gjendje "aktive" në sfond në telefon, c'ka do të thotë se llogjika e modulit punon më se miri në këto raste. Moduli është dizenjuar për të siguruar ruajtje të energjisë për klasën e trafikut në sfond, por meqenëse moduli është implementuar në kernel ai nuk bën dallim për klasat e trafiku apo ndërveprim të përdoruesit. Kjo krijon disa probleme pasi në rastet të përdoruesit në gjendje aktive ose me aplikime interaktive, UE do të mbahet në kanale DCH me së shumti duke krijuar një devijim nga puna e UE. Po ashtu në sajë të vëzhgimeve të kryera me aplikimin *tpacketcapture* e *wireshark*, në rastet e aplikimeve që përdorin protokollin TCP, moduli shton në një lloj mënyre efektin e duplikimit të paketave ACK si dhe ritransmetimeve, prandaj një konsideratë më e madhe në drejtim të protokolleve të shtresës së transportit duhet të kihet parasysh për punën në të ardhmen së bashku me efektin direkt të aplikimeve interaktive.

Të gjitha këto duhet të konsiderohen në punën tonë në të ardhmen. Gjithashtu nuk duhet të harojmë se dhe vetë vendorët e produkteve të UE apo RNC e MME-PGW duhet të tregohen më të kujdesshëm për përmirësimin e funksioneve të RRC e RLC. Një kombinim i koordinuar i zgjidhjeve eficente si në anën e UE dhe atë të rrjetit / RNC e MME-SGW/PGW do të sjellë një rritje të përfitimit energjitik në UE.

Pra puna në të ardhmen do të konsistojë:

- Tipi, qëllimi i Api si dhe protokolli i përdorur duhen studiuar më nga afër.
- Trafiku interaktiv duhet marrë në konsideratë si dhe të analizohen mundësitë e implementimit në kernel të propozimeve nga autore të ndryshëm.
- Ndryshime parametrash RLC, RRC nga operatorët mobile të update-ohen automatikisht ne filen patch kernel.
- Update OS mund të devijojnë funksionet internal te OS (si p.sh mbi netfilter) c'ka mbetet një sfidë e të ardhmës për ne për zgjidhje.
- Diskutimi në forume e më tej në lidhje me këtë propozim.

.

SHTOJCAT

Shtojca A.1 : Si të rregjistrojmë grepat

Kokat e Kernelit në Linux për Netfilter janë përfshirë poshtë direktorisë SW: */lib/modules/$(uname -r)/build/include/linux/*. Një modul Kerneli mund të rregjistrojë dhe ç'rregjistrojë një funksion duke thirrur dy funksionet e përcaktuar në dosjen/ filen *netfilter.h* si:

```
int nf_register_hook(struct nf_hook_ops *reg);
void nf_unregister_hook(struct nf_hook_ops *reg);
```

Struktura e *nf_hook_ops* mund të gjendet te *netfilter.h* dhe gjithashtu:

```
struct nf_hook_ops
{
struct list_head list;
        nf_hookfn *hook;
struct module *oëner;
        u_int8_t pf;
unsignedint hooknum;
int priority;/* Grepat janë urdhëruar sipas prioritetit rritës. */
};
```

Një shembull copëz kodi për të rregjistruar një grep jepet si më poshtë:

```
struct nf_hook_ops nfho;
nfho.hook = hook_func_in;
nfho.hooknum = NF_INET_LOCAL_IN;
nfho.pf = PF_INET;
nfho.priority = NF_IP_PRI_FIRST;
nf_register_hook(&nfho);        // Register the hook
```

Treguesi i funksionit të grepit tregon te *hook_func_in* , i cili është tip i *nf_hookfn*. Në lidhje me *netfilter.h*, ai është një tip funksioni i përcaktuar si:

```
typedefunsignedint nf_hookfn(unsignedint hooknum,
struct sk_buff *skb,
conststruct net_device *in,
conststruct net_device *out,
int (*okfn)(struct sk_buff *));
```

Kështu në kodin tonë, *hook_func_in* duhet të duket si:

```
unsignedint hook_func_in(unsignedint hooknum,
struct sk_buff *skb,
conststruct net_device *in,
conststruct net_device *out,
int (*okfn)(struct sk_buff *))
```

Fusha e mbajtësit(owner) mund të lihet dhe bosh. *PF* qëndron për familjen e protokollit. *PF_INET* është për versionin 4 te IP, ndërsa *PF_INET6* është për IP v6. Interesi ynë është për IP v4. Një

modul mund të specifikojë prioritetin e funksionit brenda grepit gjatë rregjistrimit. Prioritetet për IPv4 janë përcaktuar te *netfilter_ipv4.h :*

```
enum nf_ip_hook_priorities {
	NF_IP_PRI_FIRST = INT_MIN,
	NF_IP_PRI_CONNTRACK_DEFRAG = -400,
	NF_IP_PRI_RAW = -300,
	NF_IP_PRI_SELINUX_FIRST = -225,
	NF_IP_PRI_CONNTRACK = -200,
	NF_IP_PRI_MANGLE = -150,
	NF_IP_PRI_NAT_DST = -100,
	NF_IP_PRI_FILTER = 0,
	NF_IP_PRI_SECURITY = 50,
	NF_IP_PRI_NAT_SRC = 100,
	NF_IP_PRI_SELINUX_LAST = 225,
	NF_IP_PRI_CONNTRACK_CONFIRM = INT_MAX,
	NF_IP_PRI_LAST = INT_MAX,
};
```

Sa më e vogël vlera, aq më i lartë është prioriteti, me NF_IP_PRI_FIRST prioriteti është më i lartë. Kur grepat e Netfilter janë trigeruar, funksioni që është i rregjistruar me prioritet më të lartë thërritet i pari.

Shtojca A.2 : Funksionet e brendshme të përpunimit të paketave në protokollin IP

ip_input.c
•*ip_rcv():* ajo proceson të gjitha paketat hyrëse pa marrë parasysh destinacionin e tyre dhe porsa kontrollohet se IP header tyre është i saktë thirret më pas grepi NF_PRE_ROUTING.
•*ip_rcv_finish():* përcakton rrugën e një paketë duke thirrur te ip_route().
•*ip_local_delivery():* ajo thirret pas grepit NF_IP_LOCAL dhe nëse ka paketa unicast ose multicast për tu shpërndarë te nyja lokale.

ip_forward.c
•*ip_forward():* ajo proceson paketat me shenjën "forwarding".
•*ip_forward_finish():* ajo thirret pas përpunimit të grepit NF_IP_FORWARD dhe vendos nëse një paketë ka nevojë për copëzimin/segmentimin ose mund të kalojë direkt në fazen e daljes.
•*ip_fragment():* ai bën të gjithë punën e fragmentimit të paketës.

ip_output.c
•*ip_queue_xmit():* ajo shton kokën IP në strukturën skb_buf dhe siguron se paketa ka një tregues të vlefshëm për rrugëzimin e "cache" para se të kërkohet grepi NF_IP_LOCAL_OUT.
•*ip_queue_xmit2():* është quajtur pas përpunimit të grepit NF_IP_LOCAL_OUT dhe përcakton daljen e ndërfaqes së rrjetit me përmbajtjen e saj përkatëse të "cache".
•*ip_output():* udhëzon paketat e krijuara lokalisht te grepi NF_IP_POST_ROUTING.
•*ip_finish_output():* ajo merr paketat e dërguara dhe të gjeneruara lokalisht dhe përcakton parametrat e kokave të tyre për të hyrë te shtresa 2 dhe të thërrasë grepin NF_IP_POST_ROUTING.

•*ip_finish_output2():* ajo kalon paketën për në shtresën e 2 pas përpunimit të grepit NF_IP_POST_ROUTING.

Shtojca A.3: Aksesimi i bufferit të paketës së rrjetit

skb në funksionet e grepit lejon të aksesohet bufferi i paketës së rrjetit dhe rimarrja të informacionit të kërkuar sa më lehtë. Me probabilitet, tre informacionet më të kërkuara janë fushat e *transport_header*, *network_header* dhe *mac_hea*der përcaktuar nw*sk_buff*. Funksionet e mëposhtme janë përcaktuar në *skbuff.h* për të aksesuar këto informacione:

*staticinlineunsignedchar *skb_mac_header(conststruct sk_buff *skb);*
*staticinlineunsignedchar *skb_network_header(conststruct sk_buff *skb);*
*staticinlineunsignedchar *skb_transport_header(conststruct sk_buff *skb);*

Për shembull, nëse ne duam të marrim burimin dhe destinacionin e adresave IP të një pakete të kapur të një funksioni grepi të mësipërm, kodi më poshtë mund ta kryejë atë:

*struct iphdr *ip_header = (struct iphdr *)skb_network_header(skb);*
unsignedint src_ip = (unsignedint)ip_header->saddr;
unsignedint dest_ip = (unsignedint)ip_header->daddr;

Një gjë që duhet të përmendet është që *skb_transport_header* jo gjithmonë punon siç pritet. Ky funksion punon kur paketat e rrjetit janë procesuar nga funksione të sigurt të Netfilter i cili është te shtresa e transportit në implementimin e netfilter. Për shembull ne duam të marrim burimin dhe destinacionin e një pakete UDP. Kur një paketë del jashtë, ajo shkon fillimisht nga shtresa e aplikimit te ajo e transportit, te e rrjetit e kështu me rradhë sipas procesit të TCP/IP. Këto funksione janë ekzekutuar normalisht dhe *skb_transport_header* punon në rregull. Kodi i mëposhtëm punon si vijon:

```
struct iphdr *ip_header = (struct iphdr *)skb_network_header(skb);
struct udphdr *udp_header;
if (ip_header->protocol==17) {
   udp_header = (struct udphdr *)skb_transport_header(skb);
   src_port = (unsignedint)ntohs(udp_header->source);
   dest_port = (unsignedint)ntohs(udp_header->dest);
}
```

Kur një paketë vjen për nga kablli apo mjedis tjetër si ai ajror, ajo përshkruan nga shtresa fizike, datalinkut, rrjetit e mësipër më pas, kështu që mund të mos shkojë përmes funksioneve të përcaktuar në netfilter për *skb_transport_header* që të punojë. Kodi i mëposhtëm nuk punon në këtë rast, por aty ka një "hackim" / thyerje të rregullit që mund të kryhet si më poshtë:

```
struct iphdr *ip_header = (struct iphdr *)skb_network_header(skb);
struct udphdr *udp_header;
if (ip_header->protocol==17) {
   udp_header = (struct udphdr *)(skb_transport_header(skb)+20);
   src_port = (unsignedint)ntohs(udp_header->source);
   dest_port = (unsignedint)ntohs(udp_header->dest);
}
```

skb_transport_header aktualisht kthen një tregues te koka e adresës IP, dhe kodi anashkalon adresën IP me +20 (gjatësia tipike e një koke IP).

Shtojca B.1: Kompilimi i një moduli kernel

Për të kompiluar një modul Kernel, përdorim konfigurimin e "makefile" (duke supozuar që ruajtëm filën në *netfilter.c*) :

```
obj-m += netfilter.o
all:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
clean:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

Për të futur modulin e kernelit dhe për të hequr modulin e kernelit në versione të Linux apo Ubuntu mund të kryhet me komandat si më poshtë:

```
sudo insmod netfilter.ko
sudo rmmod netfilter
```

Shtojca B.2: Ndërtimi dhe përdorimi i libnetfilter_queue për Android

Libnetfilter_queue është një librari e hapësirës së përdoruesit duke dhënë një API të paketave që kanë qenë në pritje nga filtrimi i paketave Kernel. Është pjesa e saj që të kundërshtojë mekanizmin e vjetër të *ip_queue / libipq. libnetfilter_queue e cila* është free dhe mund të merret në linkun web më poshtë: http://netfilter.org/projects/libnetfilter_queue/. Versioni në zhvillim i libnetfilter_queue mund të aksesohet te https://git.netfilter.org/cgi-bin/ gitweb.cgi? p= libnetfilter_queue.git;a=summary.

libnetfilter_queue kërkon *libnfnetlink* dhe një Kernel që përfshin nënsistemin e *nfnetlink_queue.*

Funksione, Privilegje e Performanca

- Marrja e paketave në pritje nga nënsistemi i Kernelit *nfnetlink_queue.*
- Lëshimi i verdiktit dhe/ose ri-injektimi i paketave të ndryshuara te nënsistemi i kernelit *nfnetlink_queue.*

Duhet mundësia/opsioni i*CAP_NET_ADMIN* në mënyrë që të lejojë aplikimin tonë të marrë dhe të dërgojë paketa te hapësira e kernelit.

Për të rritur performancën e aplikimit tonë te *libnetfilter_queue*, duhet të konsiderojmë këto:

- Rrisim madhësinë default të buferit të socket me anë të *nfnl_rcvbufsiz().*
- Vendosim një vlerë të kënaqshme të proceseve tona te -20 (maximum priority).
- Vendosim ciklet e procesorit CPU në një bërthamë të lirë që nuk është përdorur për të trajtuar ndërprerjet e kartës se rrjetit (NIC).

- Vendosim opsionin NETLINK_NO_ENOBUFS te socket për të shmangur marrjen e erroreve/gabimeve ENOBUFS (kërkon Linux kernel version >= 2.6.30).
- Shikojmë opsionin --queue-balance në NF_QUEUE target për aplikimet multi-threaded (kjo kërkon Linux kernel >= 2.6.31).

Pikësëpari duhet të shikojmë kernelin e Androidit nëse është kompiluar me suportin për *libentfilter_queue*. Për këtë:

- lidhim pajisjen UE Android në një PC me OS Ubuntu ver 14_4.

- fusim cmnd "*adb pull /proc/config.gz*" për të nxjerrë filën *config.gz* nga pajisja UE.

- bëjmë ekstrakt filën *config.gz* dhe do të marrim një file të emërtuar si *config*. Kjo është aktualisht fila e konfigurimit te Kernelit Linux te Androidit.

- kërkojmë për CONFIG_NETFILTER_ADVANCED, CONFIG_NETFITLER_NETLINK dhe CONFIG_NETFILTER_NETLINK_QUEUE në filën e *config* dhe sigurohemi që ato nuk janë komentuar jashtë. Nëse SW/build i UE Android nuk është kompiluar me këto "karakteristika", do të duhet të përpilojmë një SW të përshtatur/ specifikur për të përdorur *libnetfilter_queue*.

Së dyti, duhet të rootojmë/rrugëzojmë pajisjen Android (të drejtat për të ndërhyrë në sistemin OS pa kufizime).

Se treti, duhet të sigurohemi që pajisja ka një program për iptables. Programi iptabels është përdorur për të konfiguruar tabelën e filtrimit të paketave kernel. Jep cmnd si më poshtë për të parë:

adb shell ,
su - ,
iptables –list

Nëse terminali nuk nxjerr ndonjë error rreth programeve jo të gjetura, atëherë kemi iptables të instaluar. Nëse nuk e kemi duhet të instalojmë në pajisjen UE Android një aplikim *busybox*, ose të përpilojmë programin vetjak të iptables. Sikurse është parë, kodi i burimit Android-i përfshin iptables në folderin e jashtëm.

Menaxhimi i pritjeve

*int nfq_fd (struct nfq_handle *h)*
*struct nfq_q_handle **
*nfq_create_queue (struct nfq_handle *h, u_int16_t num, nfq_callback *cb, void *data)*
*int nfq_destroy_queue (struct nfq_q_handle *qh)*
*int nfq_handle_packet (struct nfq_handle *h, char *buf, int len)*
*int nfq_set_mode (struct nfq_q_handle *qh, u_int8_t mode, u_int32_t range)*
*int nfq_set_queue_maxlen (struct nfq_q_handle *qh, u_int32_t queuelen)*
*int nfq_set_verdict (struct nfq_q_handle *qh, u_int32_t id, u_int32_t verdict, u_int32_t data_len, const unsigned char *buf)*
*int nfq_set_verdict2 (struct nfq_q_handle *qh, u_int32_t id, u_int32_t verdict, u_int32_t mark, u_int32_t data_len, const unsigned char *buf)*

*int nfq_set_verdict_mark (struct nfq_q_handle *qh, u_int32_t id, u_int32_t verdict, u_int32_t mark, u_int32_t data_len, const unsigned char *buf)*

Përshkrimi

Porsa libraria *libnetfilter_queue* është inicuar, është e mundur të lidhim (bind) programin te një pritje (queue) specifike. Kjo mund të arrihet duke përdorur nfq_create_queue(). Kjo pritje (queue) më pas mund të kthehet nëpërmjet nfq_set_mode() ose nfq_set_queue_maxlen().

Një pjesë kodi që krijon një pritje (queue) të numëruar 0:

```
printf("binding this socket to queue '0'\n");
        qh = nfq_create_queue(h,  0, &cb, NULL);
        if (!qh) {
                fprintf(stderr, "error during nfq_create_queue()\n");
                exit(1);
        }
        printf("setting copy_packet mode\n");
        if (nfq_set_mode(qh, NFQNL_COPY_PACKET, 0xffff) < 0) {
                fprintf(stderr, "can't set packet_copy mode\n");
                exit(1);
        }
```

Hapi tjetër është mbartja (handle) e paketave hyrëse e cila mund të bëhet me një lak të mbyllur loop:

```
fd = nfq_fd(h);

        while ((rv = recv(fd, buf, sizeof(buf), 0)) >= 0) {
                printf("pkt received\n");
                nfq_handle_packet(h, buf, rv);
```

Kur vendimi në paketë është marrë, verdikti duhet të jepet duke thërritur nfq_set_verdict()ose nfq_set_verdict2(). Verdikti përcakton fatin e paketës si vijon në grepat: NF_ACCEPT, NF_DROP, NF_STOLEN, NF_QUEUE, NF_REPEAT, NF_STOP. Të dhënat dhe informacioni rreth paketës mund të nxirren duke përdorur funksionin e mesazhit analizues (parsing).

Shtojca B.3: Mekanizmi i bllokimit kernel

Mekanizmat e kyçjes të Kernel janë të nevojshme kur sistemi në mënyrë paralele kryen/ mban shumë mbajtësa që bëjnë "share" të njëjtat të dhëna ose burime. Ekzekutimi paralel mund të prodhojë një gjendje gare ku shumë mbajtës tentojnë të aksesojnë një burim të përbashkët /"share" në të njëjtën kohë. Funksioni i mekanizmit të kyçjes/bllokimit është të mbrojë përkundër akseseve të shumëfishta te pjesët e kodit ku burimi share / i përbashkët ndodhet, i cili është quajtur seksioni kritik. Moduli është formuar nga tre mbajtës kryesorë të cilët ndajnë disa struktura të dhënash. Seksione kritik të modulit janë:

- Seksioni ku paketat janë në rradhë/queue: Pritësit/queue janë të aksesuar nga mbajtësi *Main* dhe *Reinject* për tu vënë në pritje dhe transmetuar;
- Seksioni ku gjendja e UE është e updatuar: Variabli i gjendjes së UE përcakton gjendjen e UE dhe ai është i aksesuar nga *mbajtësi i kotrollit të gjendjeve* të *UE* dhe *Reinject*;

- Seksioni ku mbajtësi i Reinject është “zgjuar”: Variabli reinject_sleep tregon nëse mbajtësi reinject fle ose jo. Mbajtësi reinject është në gjumë derisa variabli reinject/zgjim është vënë në statusin on dhe kështu ai zgjohet. Këto variabla janë të aksesueshëm nga mbajtësit Main dhe Reinject.

Në mënyrë që të mbrojmë këto seksione kritike ne duhet të implementojmë mekanizmin më të përshtatshëm të bllokimit. Me poshtë përshkruajmë mekanizma të ndryshëm dhe përdorimet e rekomanduara të tyre [108]:

- Spinlocks: Procesi në mënyrë të vazhdueshme tenton të kërkojë bllokimin deri sa është e mundur (mban për një periudhe të shkurtër)
- Semaphores: Procesi fle derisa bllokimi është i mundur (mban per nje periudhe te gjate).
- Reader-Writer locks: Eshtë një tip i spinlock i cili lejon disa procese të kërkojnë leximin e bllokimit, por vetëm një process ka të drejtën e të shkruarit.

Eshtë përdorur bllokimi i tretë në listë sepse koha e bllokim-mbajtjes në seksionin kritik tonin pritet të jetë e shkurtër (ms) dhe mbajtësa të shumëfishtë aksesojnë zonat e mbrojtura për të lexuar dhe shkruar të dhënat.

Shtojca B.4: Moduli Android për planifikim transmetimi paketash në një komunikim mobile të dhënash

```
#include "/home/lruci/smartphone/ossw/include/linux/Kernel.h" /* therittet gjithmone */
#include "/home/lruci/smartphone/ossw/include/linux/module.h" /* therittet gjithmone */
#include "/home/lruci/smartphone/ossw/include/linux/moduleparam.h"
#include "/home/lruci/smartphone/ossw/include/linux/time.h"
#include "/home/lruci/smartphone/ossw/include/linux/types.h"
#include "/home/lruci/smartphone/ossw/include/linux/timer.h"
#include "/home/lruci/smartphone/ossw/include/linux/string.h"
#include "/home/lruci/smartphone/ossw/include/linux/netfilter.h"
#include "/home/lruci/smartphone/ossw/include/linux/netfilter_ipv4.h"
#include "/home/lruci/smartphone/ossw/include/linux/delay.h"
#include "/home/lruci/smartphone/ossw/include/linux/wait.h"
#include "/home/lruci/smartphone/ossw/include/linux/kthread.h"
#include "/home/lruci/smartphone/ossw/include/linux/spinlock.h"
#include "/home/lruci/smartphone/ossw/include/net/netfilter/nf_queue.h"
#include "/home/lruci/smartphone/ossw/include/net/ip.h"
#define DRIVER_AUTHOR "lruci <luan.rucci@gmail.com>"
#define DRIVER_DESC "Modul Android për eficencë energjie në 3G/4G "

#ifdef “KERNEL”
#ifdef “CONFIG_NETFILTER”

const unsigned int PCH=1;        /* deklarim kanalet 3G/4G  */
const unsigned int FACH=2;
const unsigned int DCH=3;
const unsigned int Pm=1;         /* Deklarim kushtet e tranzitimit, pragjet  */
const unsigned int Pv=2;
const unsigned int Pv_FACH=3;
const unsigned int sdeadline_Pm=1; /* afatet e ruajtjes së paketave */
const unsigned int sdeadline_Pv=2;
static unsigned int sdeadline;
static unsigned int uestate_current;
static unsigned int uestate_last;
static struct nf_hook_ops nfho;   /*strukturë që mban një grup të funksioneve të grepave  */
static struct nf_queue_handler nfqh;
static DEFINE_RWLOCK(queue_lock);
static LIST_HEAD(PM);
static LIST_HEAD(PV);
static unsigned int Pm_ptotal;
static unsigned int PV_ptotal;
```

```
static struct task_struct *reinject_task;
static struct task_struct *uestate_control_task;
static unsignedlong int gtnow;
static unsignedlong int gtlast;
const unsigned long int T1=2;    /*sek, Kohët e pasivitetit (DCH->FACH e FACH ne PCH ) */
const unsigned long int T2=5;
static unsigned int B1=530;   /* Pragjet e memorjeve te protokollit RLC ne Byte*/
static unsigned int B2=290;
static unsigned int Kp=300;   /* ms, koha për fshirjen e bufferit RLC pas transmetimit */
static unsigned int Kv=60;   /* sek,Dritare minimale e vonesës se transferimeve , 90 e 120 sek */

static unsigned int sum=0;
wait_queue_head_t reinject_wqh;
static int reinject_wakeup=0;
static int packet_tx=0;
static bool reinject_sleep=true;

module_param(B1, int,0);            /* Buferat RLC  */
module_param(B2, int, 0);
module_param(Kp, int,0);
module_param(uestate_current, int,0);
module_param(Kv, int,0);

static uint16_t swap_uint16( uint16_t val )
{
return (val <<8) | (val >>8 );
}

                                /* funksion që thërritet nga Grepi */
unsigned int hook_func(unsigned int hooknum, struct sk_buff *skb, const struct net_device *in,
const struct net_device *out, int (*okfn)(struct sk_buff *))
{
          struct timespec tspec;
          struct iphdr *network_header; /* deklarim i leximit të paketave të header-it */
          network_header = (struct iphdr *)skb_network_header(skb);
          if((network_header->protocol==1)||(strcmp(skb->dev->name,"lo")==0))
          {
                    return NF_DROP;
          }
          else
          {
                    if(uestate_current==DCH) /* Kontroll nese UE eshte ne kanalin DCH */
                    {
                              getnstimeofday(&tspec);
                              write_lock_bh(&queue_lock);
                              gtlast=tspec.tv_sec;
                              write_unlock_bh(&queue_lock);
                              return NF_ACCEPT;
                    }
                    else
                    {
                              return NF_QUEUE;
                    }
          }
}

static void enqueue_packet(struct nf_queue_entry *entry)
{
          if((entry->skb->len)>=B2)    /* nëse madhësia e paketës e me madhe se kufiri B2 */
          {                            /* paketa vihet në rradhë në PM*/
                    list_add_tail(&entry->list, &PM);
                    sdeadline=sdeadline_PM;
          }
          Else                         /* në të kundërt vihet në Pv */
          {
                    list_add_tail(&entry->list, &Pv);
                    sdeadline=sdeadline_Pv;
          }
}

static int enqueue(struct nf_queue_entry *entry,unsigned int queuenum)
{

          enqueue_packet(entry);           /* vendos paketat në pritje */
          if(reinject_sleep)               /* zgjon mbajtësin reinject në gjumë */
          {
```

```
                    write_lock_bh(&queue_lock);
                    reinject_wakeup=1;
                    reinject_sleep=false;
                    wake_up_interruptible(&reinject_wqh);
                    write_unlock_bh(&queue_lock);
          }
          return1;
}

static void reinject_queue(int queue_list)              /* vendosja ne pritje */
{
          struct nf_queue_entry *entry, *next;
          struct timespec tspec;
          unsigned int npackets=1;
          if(queue_list==PM)                /* nese jane paketat e medha kontrollo gjendjen*/
          {
                    list_for_each_entry_safe(entry, next, &PM, list)
                    {
                              list_del(&entry->list);
                              if(entry && entry->skb)
                              {
                                        nf_reinject(entry, NF_ACCEPT);
                                        getnstimeofday(&tspec);
                                        write_lock_bh(&queue_lock);
                                        gtlast=tspec.tv_sec;
                                        write_unlock_bh(&queue_lock);
                              }
                    }
          }
          else if(queue_list==Pv)           /* nese jane paketat e vogla kontrollo gjendjen */
          {
                    list_for_each_entry_safe(entry, next, &Pv, list)
                    {
                              list_del(&entry->list);
                              if(entry && entry->skb)
                              {
                                        nf_reinject(entry, NF_ACCEPT);
                                        getnstimeofday(&tspec);
                                        write_lock_bh(&queue_lock);
                                        gtlast=tspec.tv_sec;
                                        write_unlock_bh(&queue_lock);
                              }
                    }
          }
}

static void wait_time(unsigned long int time)       /* vendos kohe vonese */

{
          ssleep(time);
}

static void wait_time_ms(unsigned long int time)
{
          msleep(time);
}

static unsigned int channelChoice (unsigned int vs)
{
          unsigned int channel;
          //unsigned int pDCH=700;    /*Konsumi fuqisë në kanalin DCH në mW */
          //unsigned int pFACH=400;    /*Konsumi fuqisë në kanalin FACH në mW */
          unsigned int limit=2634;  /* pDCH/pFACH*B2*T1/Kp*1000; /*
          if (vs<limit)    /*((pDCH*Kp)/(pFACH*B2*T1*1000))) ku Kp është në ms adhe shumëz me 1000
          {
                    channel=FACH;
          }
          else
          {
                    channel=DCH;
          }
          return channel;
}

int reinject_thread(void *unused1)
{
```

```
          unsigned int vl,vs,channel;     /* volumet Vl e Vs shuma e bufferave Pm e Pv */
          struct nf_queue_entry *entry, *next;
          struct nf_queue_entry *first_entry=NULL;
          struct timespec tspec;
          while(1)
        {
                    reinject_sleep=true;        /* ne pergjumje = jo transmetim */
                    wait_event_interruptible(reinject_wqh,reinject_wakeup==1);
                    write_lock_bh(&queue_lock);
                    reinject_wakeup=0;
                    reinject_sleep=false;
                    write_unlock_bh(&queue_lock);
                    sum=0;
                    vl=0;
                    vs=0;

                    while((!list_empty(&PM))||(!list_empty(&Pv)))
                    {
                              if(uestate_current==DCH)  /* nëse në DCH transmeton gjithë PM e PV*/
                              {
                                        reinject_queue(PM);
                                        reinject_queue(Pv);
                              }
                              else if(uestate_current==FACH)  /* nëse në FACH kontrollo paketat
hyrëse dhe shumën në buffer pas çdo kalimi pakete */
                              {
                                        if((!list_empty(&PM))&&(list_empty(&Pv)))
                                                  msleep(3);
                                        while(!list_empty(&Pv))   /* transmeto paketat në Pv  1 nga
1 që të mos kemi kalim në DCH  */
                                        {
                                                  first_entry=list_first_entry(&Pv,struct
nf_queue_entry,list);
                                                  sum=sum+(first_entry->skb->len);
                                                  list_del_init(&first_entry->list);
                                                  if(sum>B2  /* nëse shuma më e madhe se kalimi
FACH në DCH prit për një kohë Kp */
                                                  {
                                                            wait_time_ms(Kp);
                                                            sum=first_entry->skb->len;
                                                  }
                                                  print_packet_info(first_entry,1);
                                                  nf_reinject(first_entry,NF_ACCEPT);  /* ri-
injekto paketat */
                                                  getnstimeofday(&tspec);
                                                  write_lock_bh(&queue_lock);
                                                  gtlast=tspec.tv_sec;
                                                  write_unlock_bh(&queue_lock);

                              }
                              else if(uestate_current==PCH)  /* nëse në PCH njofto dhe prit për
kohën Kv */
                              {
                                        printk(KERN_ALERT "Ne PCH\n");

                                        if(sdeadline==sdeadline_PM) /* nëse ka aritur afati kohor
transmeto të dhënat në PM dhe kalon në DCH */
                                        {
                                                  wait_time(Kv);
                                                  reinject_queue(PM);
                                                  getnstimeofday(&tspec);
                                                  write_lock_bh(&queue_lock);
                                                  gtlast=tspec.tv_sec;
                                                  uestate_current=DCH;
                                                  write_unlock_bh(&queue_lock);
                                        }
                                        else if(sdeadline==sdeadline_Pv)   /* nëse ka arritur afati
kohë transmeto llogaritjet Vl për çdo paketë hyrëse */
                                        {
                                                  wait_time(Kv);
                                                  vl=0;
                                                  printk(KERN_ALERT "LLOGARITJE E Vl\n");
                                                  list_for_each_entry_safe(entry, next, &PM, list)
                                                  {
                                                            vl=vl+entry->skb->len;
                                                  }
```

```
                                                if(vl>=B1)  /* nëse Vl > bufferi i kalimit DCH,
tranmeto PM e kalo në DCH */
                                                {
                                                          reinject_queue(PM);
                                                          getnstimeofday(&tspec);
                                                          write_lock_bh(&queue_lock);
                                                          gtlast=tspec.tv_sec;
                                                          uestate_current=DCH;
                                                          write_unlock_bh(&queue_lock);
                                                }
                                                else /* përndryshe llogarit Vs */
                                                {
                                                          vs=0;
                                                          list_for_each_entry_safe(entry, next,
&Pv, list)
                                                          {
                                                          vs=vs+entry->skb->len;
                                                          }
                                                          channel=channelChoice(vs);
                                                          if(channel==DCH)    /* nese ne DCH
transmeto PV e PM, */
                                                          {
                                                          reinject_queue(Pv);
                                                          reinject_queue(PM);
                                                          getnstimeofday(&tspec);
                                                          write_lock_bh(&queue_lock);
                                                          gtlast=tspec.tv_sec;
                                                          uestate_current=DCH;
                                                             write_unlock_bh(&queue_lock);
                                                          }
                                                          else /* përndryshe transmeto PM, prit
një kohë Kp e kalon në FACH */
                                                          {
                                                          reinject_queue(PM);
                                                          wait_time_ms(Kp);
                                                          getnstimeofday(&tspec);
                                                          write_lock_bh(&queue_lock);
                                                          gtlast=tspec.tv_sec;
                                                          uestate_current=FACH;
                                                          write_unlock_bh(&queue_lock);
                                                          }
                                                }
                                       }
                              }
                    }
          }
          return 0;
}

int uestate_control_thread(void *unused1)     /* kontroll i kanalit të UE */
{
           struct timespec tspec;
           while(1)
          {
                    msleep(3);
                    uestate_last=uestate_current;

                    getnstimeofday(&tspec);
                    gtnow=tspec.tv_sec;

                    if((uestate_last==DCH)&&((gtnow-gtlast)>=(T1+T2)))     /* nëse UE ka qenë në
DCH dhe ka kaluar koha T1+T2  */
                    {
                              write_lock_bh(&queue_lock);
                              uestate_current=PCH;  /* UE është në PCH */
                              write_unlock_bh(&queue_lock);
                    }
                    else if((uestate_last==DCH)&&((gtnow-gtlast)>=T1)&&((gtnow-gtlast)<=(T1+T2)))
/* nëse ka qenë në DCH dhe koha më e madhe se T1 e më e vogël se T1+T2 */
                    {
                              write_lock_bh(&queue_lock);
                              uestate_current=FACH;        /* UE është në FACH */
                              gtlast=gtnow;
                              write_unlock_bh(&queue_lock);
                    }
```

```
                    else if((uestate_last==FACH)&&((gtnow-gtlast)>=T2))  /* nëse UE ka qenë në
FACH dhe ka kaluar koha T2 */
                    {
                              write_lock_bh(&queue_lock);
                              uestate_current=PCH;         /* UE është në PCH */
                              write_unlock_bh(&queue_lock);
                    }
          }
return0;
}

staticvoid init_uestate(unsigned int state)
{
          if(state==DCH)
          {
                    uestate_current=DCH;
                    uestate_last=uestate_current;
          }
          else if(state==FACH)
          {
                    uestate_current=FACH;
                    uestate_last=uestate_current;
          }
          else if(state==PCH)
          {
                    uestate_current=PCH;
                    uestate_last=uestate_current;
          }
}

                                        /*Thirrur kur moduli ngarkohet duke përdorur 'insmod' */
int init_module()
{
          int status_qhandler;
          printk("Inicializim Moduli Netfilter \n");

          nfqh.name="cola1";
          nfqh.outfn=&encolar;

          status_qhandler=nf_register_queue_handler(PF_INET,&nfqh);

          if(status_qhandler<0)
          {
                    printk(KERN_INFO "\n~~~Nuk rregjistron dot modulin~~~\n");
                    return0;
          }
          else
          {
                    printk(KERN_INFO "\n~~~~Rregjistrim i modulit~~~~\n");

                    nfho.hook = hook_func;     /*Funksion që thirret kur kushtet më poshtë
përmbushen */
                    nfho.hooknum = NF_INET_POST_ROUTING;  //i thirrur direkt pasi paketa është
marrë, grepi i parë në Netfilter */
                    nfho.pf = PF_INET;    /* paketat IPV4  */
                    nfho.priority = NF_IP_PRI_FIRST;    /* vendos në prioritetitn me të lartë
përmbi të gjitha funksionet e grepit */
                    nf_register_hook(&nfho);    /* regjistro grepin */

                    packet_tx=0;
                    init_uestate(PCH);
                    init_waitqueue_head(&reinject_wqh);


          uestate_control_task=kthread_run(uestate_control_thread,NULL,"UESTATE_CONTROL_THREAD");
                    reinject_task=kthread_run(reinject_thread,NULL,"REINJECT_THREAD");

                    return0;
          }
}

                                  /* Thirrur kur moduli hiqet duke përdorur 'rmmod' */
void cleanup_module()
{
  printk("Heqja e modulit Netfilter\n");
```

```
  kthread_stop(reinject_task);
  kthread_stop(uestate_control_task);

  nf_unregister_hook(&nfho);

  nf_unregister_queue_handler(PF_INET,&nfqh);
}

MODULE_LICENSE("GPL");
MODULE_AUTHOR(DRIVER_AUTHOR);          /* Kush e shkruajti këtë modul? */
MODULE_DESCRIPTION(DRIVER_DESC);       /* Çfarë bën ky modul? */

#endif // "CONFIG_NETFILTER"
#endif // "KERNEL"
```

REFERENCAT

[1] G.Paolo Perrucci, Frank H.P. Fitzeky, G.Sassoy, Wo.Kellererx and J.Widmerx: "*On the Impact of 2G and 3G Network Usage for wireless Phones' Battery Life*", European Wireless 2009, pg. 255-259;

[2] Ofversten J.*: "Mobile Internet Battery Life. Challenges for application SW development*", Forum Nokia webinar. 16.12.2009. Cited 1.12.2010. Url:http://www.forum.nokia.com/Library/Multimedia/Webinars.xhtml;

[3] F. Qian, Z. Wang, A. Gerber, Z. Mao, S. Sen and O. Spatscheck: "*Profiling resource usage for mobile applications: a cross-layer approach",* in the Proceedings of the 9th international conference on Mobile systems, applications, and services (MobiSys), 2011;

[4] Y.Wang, X.Liu,C.H.Hsu. Update: "*User-Profile-Driven Adaptive Transfer"*, Journal , ACM Transactions on Embedded Computing Systems (TECS), Volume 15 Issue 3, July 2016, ACM New. York, NY, USA;

[5] G.P.Perrucci, Frank H.P. Fitzek, G.Sasso, M.Katz. "*Energy Saving Strategies for Wireless Devices using Wake-up Signals",* in the 4th International Mobile Multimedia Communications conference, MobiMedia 2008. ICTS/ACM, July 2008;

[6] N.Jacobsson, M.Lopes, Sh.Islam. *"Energy Consumption In Wireless Networks Progress Report"*.

[7] N. Balasubramanian, A. Balasubramanian, and A.Venkataramani: *"Energy consumption in wireless phones: a measurement study and implications for network applications,"* in Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference, ser. IMC '09 , pg. 280–293. New York, NY, USA: ACM, 2009;

[8] M.Siekkinen, M. A.Hoque, J. K.Nurminen, and M.Aalto. Streaming over 3G and LTE: "*How to save smartphone energy in radio access network-friendly way"*, in Proceedings of the 5th Workshop on Mobile Video, MoVid '13, pg. 13-18. ACM, 2013;

[9] **L.Ruci**, L.Karcanaj, *"Evoluimi i teknologjisë mobile dhe i shërbimeve të internetit, zgjidhja e problematikave",* Tiranë, Shqipëri, Korrik 06-07, 2012 në revistën "Hulumtime Shkencore", UKT, me ISSN 2226-1605;

[10] E.Jung, Y.Wang, I.Prilepov, *PACE Yourself: "Power Aware Collaborative Execution on wireless Phones"*. 2010;

[11] Jurvansuu, Jarmo Prokkola, Mikko Hanski and Pekka Perälä; "*HSDPA Performance in Live Networks"*, Marko, publication in the ICC 2007;

[12] Nokia Siemens Solutions Smart_Lab_WhitePaper, *www.nsn.com/system/files/.../Smart_Lab_WhitePaper_27012011*, low-res.pdf;

[13] Smartphone Solutions White Paper. www.huawei.com/ilink/en/download/HW_193034;

[14] Cisco. Cisco Visual Networking Index: "*Forecast and Methodology"*, 2012-2017, May 2013, http://tinyurl.com/mh7t5gx. Accessed 12.6.2013;

[15] Behaviour Analysis of Smartphone. www.huawei.com/en/static/hw-001545.pdf;

[16] C.-C. Lee, J.-H. Yeh, and J.-C. Chen. "*Impact of inactivity timer on energy consumption in WCDMA and cdma 2000"*, in Wireless Telecommunications Symposium, pg. 15-24, IEEE. 2004;

[17] http://tabtimes.com/feature/ittech-developers/2013/01/24/app-quality-alliance-developers-can- do-more-improve-their;

[18] http://tabtimes.com/feature/ittech-developers/2013/02/27/esential-10-key-tips-developers-smartphone-and-tablet-apps;

[19] http://apigee.com/about/pressrelease/apigee-survey-users-reveal-top-frustrations-lead-bad-wireless-app-reviews;

[20] Smartphone usage experience, summary report, January 2013, www.ericsson.com/res/ docs/. ../smartphone-usage-experience-report.pdf;

[21] T.Pering, Y.Agarwal, R.Gupta, and R. Want. CoolSpots: "*Reducing the Power Consumption of Wireless Mobile Devices with Multiple Radio Interfaces*", in Proceedings of the 4th international conference on Mobile systems, applications and services, June 2006, pg. 220-232;

[22] Forum Nokia webinar 16.12.2009, cited 1.12.2010. http://www.forum.nokia.com/Library /Multimedia/ Webinars.xhtml;

[23] A. Schulman, V. Navda, R. Ramjee, N. Spring, P. Deshpande, C. Grunewald, K. Jain, and V. N. Padmanabhan. Bartendr: "*A practical approach to energy-aware cellular data scheduling*", in Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking, MobiCom '10, pg. 85-96. ACM, 2010;

[24] Netmarketshare web, http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd;

[25] B.Walke, P.Seidenberg, M.P.Althoff. "*UMTS: the fundamentals"* Aachen University, Germany, 2001;

[26] http://www.thingsthatscale.com/2010/03/10-issues-with-smartphone-apps//;

[27] Nurminen J., "*Parallel connections and their effect to battery consumption of mobile phone"*, in Proceedings of the 7th IEEE Consumer Communications & Networking Conference, January 2010;

[28] M. Kjaergaard, J.Langdal, and T.Godsk. "*Demonstrating EnTracked a System for Energy-Efficient Position Tracking for Mobile Devices"*, in Proceedings of the 12th ACM international conference adjunct papers on Ubiquitous computing (September 2010), pg. 367-268;

[29] H. Liu, Y. Zhang, and Y. Zhou. TailTheft: *"Leveraging the wasted time for saving energy in cellular communications",* in Proceedings of the 6th International Workshop on MobiArch, MobiArch '11, pg. 31-36. ACM, 2011;

[30] https://whitequark.org/blog/2011/09/12/tweaking-linux-tcp-stack-for-lossy-wireless-networks/

[31] A. Mungur, M. Dunmore and C. Edwards. "*Design and Implementation of the GSE/8+8 Solution"*, 2009;

[32] Jochen H. Schiller, "Wireless communications", 2nd edition London, 2003;

[33] Tarmo Anttalainen, *"Introduction to Telecommunications Network Engineering"*, 2nd edition, Boston, 2003;

[34] C.-C. Lee, J.-H. Yeh, and J.-C. Chen, *"Impact of inactivity timer on energy consumption in WCDMA and cdma2000*", in Wireless Telecommunications Symposium, pg. 15-24, IEEE. 2004;

[35] M. Calder and M.K. Marina. "*Batch Scheduling of Recurrent Applications for Energy Savings on Mobile Phones"*, in 7th Annual IEEE Communications Society Conference on Sensor Mesh and Ad Hoc Communications and Networks (SECON), pg. 1-3, June 2010;

[36] V. K¨on¨onen and P. Paakkonen. "*Optimizing power consumption of always-on applications based on timer alignment"*, in the 4th IFIP International Conference on New Technologies, Mobility and Security (NTMS), 2011;

[37] Carroll and G. Heiser: "*An analysis of power consumption in a smartphone*", in Proceedings of the 2010 USENIX conference on USENIX annual technical conference, ser. USENIXATC'10. Berkeley, CA, USA: USENIX Association, 2010, pg. 21–21;

[38] Rice and S. Hay: *"Measuring wireless phone energy consumption for 802.11 wireless networking"*, in *Pervasive Computing and Communications (PerCom), 2010;*

[39] Rice and S. Hay: "*Decomposing power measurements for wireless devices",* in Pervasive Computing and Communications (PerCom), 2010 IEEE International Conference, April 2 2010, pg. 70 –78;

[40] **L. Ruci**, L. Karcanaj, O. Shurdi, *"Smartphones' Influence and challenges for end users and Mobile operators"*, www.ijsint.org, Vol. 1 No. 7, July 2013 (International Journal of Science Innovation new Technologies) pg. 55—60, ISSN: 2223-2257 (Print), 2225 – 0751 (online);

[41] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck: "*A close examination of performance and power characteristics of 4G LTE networks*, in Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services, MobiSys '12, pg. 225-238. ACM, 2012;

[42] M. Pedram, M. Chang, H. Shim. "*A Compressed Frame Buffer to Reduce Display Power Consumption in Mobile Systems"*, Asia and South Pacific Design Automation Conference (2004), Pacifico Yokohama, Yokohama, Japan, Jan. 27-30, 2004, pg. 818-823;

[43] Barr, K. C. and Asanovic, "*K. Energy-aware lossless data compression. ACM Transactions on Computer Systems"*, August 2006, vol. 24, no. 3, pg. 250-291;

[44] R.Maddah, S.Sharafeddine. *"Energy-Aware Adaptive Compression Scheme for Mobile-to-Mobile Communications,* IEEE 10th International Symposium on Spread Spectrum Techniques and Applications, August 2008, pg. 688-691;

[45] L. Wang and J. Manner. "*Evaluation of data compression for energy aware communication in mobile networks"*, in International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC '09, pg. 69-76. IEEE, Oct. 2009;

[46] N. Ding, D. Wagner, X. Chen, A. Pathak, Y. C. Hu, and A. Rice. *"Characterizing and modeling the impact of wireless signal strength on smartphone battery drain*", in Proceedings of the ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS '13, pg. 29-40. ACM, 2013;

[47] A. Cotte, A. Meyerson, and B. Tagiku. "*Energy-efficient mobile data transport via online multi-network packet scheduling. Sustainable Computing: Informatics and Systems*", 1(3): pg. 196 - 212, 2011.

[48] J. Huang, F. Qian, Z. M. Mao, S. Sen, and O. Spatscheck. *"Screen off traffic characterization and optimization in 3G/4G networks*",in Proceedings of the 2012 ACM Internet Measurement Conference, IMC'12, pg. 357-364. ACM, 2012;

[49] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck., "*Characterizing radio resource allocation for 3G networks*", in Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, IMC '10, pg. 137-150. ACM, 2010;

[50] T. Oliveira, E. Ursini, and V. Timoteo. "*Simulation inspired model for energy consumption in 3G always-on mobiles*", in IEEE 2nd National Conference on Telecommunications (CONATEL), pages 1-7, 2011;

[51] Sh. Deng, H. Balakrishnan, "*Traffic-aware techniques to reduce 3G/LTE wireless energy consumption',* in proceeding CoNEXT '12. pg. 181-192 ACM New York, NY, USA ©2012;

[52] R. Kemp, N. Palmer, T. Kielmann and H. Bal. Cuckoo: *"A Computation Offloading Framework for Smartphones".* MobiCASE, October 2010;

[53] A. Roy, S. M. Rumble, R. Stutsman, P. Levis, D. Mazi_eres, and N. Zeldovich. *"Energy Management in Mobile Devices with the Cinder Operating System",* in Proceedings of the 6th Conference on Computer Systems, EuroSys '11, pg. 139-152. ACM, 2011;

[54] N. Jacobsson, M. Lopes, Sh. Islam. "*Energy Consumption In Wireless Networks"*, *Progress Report;*

[55] E. J. Vergara and S. Nadjm-Tehrani. Energy-aware cross-layer burst buffering for wireless communication. In 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet (e- Energy), 2012;

[56] A. Abdelmotalib, Zh. Wu: "*Power Consumption in Smartphones (Hardware Behaviourism)"*, in IJCSI International Journal of Computer Science Issues, ISSN (Online): 1694-0814, Vol. 9, Issue 3, No 3, May 2012;

[57] Rice and S. Hay: "*Measuring wireless phone energy consumption for 802.11 wireless networking, in Pervasive Computing and Communications",* article in Press, www.elsevier.com/locate/pmc, July 2010;

[58] R. Murmuria, J. Medsger, A. Stavrou and J. M. Voas: "*Wireless Application and Device Power Usage Measurements"*, in SERE 2012.

[59] H. Haverinen, J. Siren and P. Eronen: *"Energy Consumption of Always-On Applications in WCDMA Networks",* in 2007;

[60] H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan, and D. Estrin. "*Diversity in smartphone usage"*, in Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, MobiSys '10, pg. 179-194. ACM, 2010;

[61] A. Barbuzzi, F. Ricciato, and G. Boggia. *"Discovering parameter setting in 3G networks via active measurements",* Communications Letters, IEEE, 12(10): pg. 730-732, 2008;

[62] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra and P. Bahl. MAUI: "*Making smartphones last longer with code offload"*, in Proceedings of the 8th international conference on Mobile systems, applications, and services (MobiSys), 2010;

[63] I. Puustinen and J. Nurminen: "*The effect of unwanted internet traffic on cellular phone energy consumption*",in the 4th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2011, pg. 1-5, IEEE. Feb. 2011;

[64] P. Perala, A. Barbuzzi, G. Boggia, and K. Pentikousis: *"Theory and practice of RRC state transitions in UMTS networks"*, in GLOBECOM Workshops, 2009, pg. 1-6, IEEE. 2009;

[65] R. Mittal, A. Kansal, and R. Chandra: *"Empowering developer's to estimate app energy consumption"*, in Proceedings of the 18th Annual International Conference on Mobile Computing and Networking, Mobicom'12, pg. 317-328. ACM, 2012;

[66] M. Pedersen, F. Fitzek, G. P. Perrucci, and T. Larsen. Energy and link measurements for mobile phones using IEEE 802.11b/g, in Proceedings of the 6th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks and Workshops, WiOPT 2008, pages 36-42, IEEE, April 2008;

[67] **L. Ruci,** L. Karcanaj, O. Shurdi: "*Nokia Windows Mobile's Power Consumption Measurements and Analysis"*. ICT Innovations 2014, Web Proceedings ISSN 1857-7288 © ICT ACT, 2014, 6th ICT Innovations Conference, (ICT ACT http://ict-act.org, pg. 1-12, Skopje 2014, On line edition published on http://ictinnovations.org, 9-12 September, Ohrid, Macedonia;

[68] E. J. Vergara, S. N.Tehrani, M. Asplund, and U. Zurutuza. "*Resource footprint of a many cast protocol implementation on multiple mobile platforms"*, in Proceedings of the 2011, 5th International Conference on Next Generation Mobile Applications, Services and Technologies, NGMAST '11, pg. 154-160. IEEE, 2011;

[69] E. J. Vergara, J. Sanjuan, and S. Nadjm-Tehrani, *"KLS energy-efficient 3G background traffic shaper for android smartphones,"* 2013, 9th Int. Wirel. Commun. Mob. Comput. Conf., pg. 443–449, Jul. 2013;

[70] Z. Wang, A. Gerber, Z.M. Mao, S. Sen and O. Spatscheck: *"Characterizing radio resource allocation for 3G networks"*, in Proceedings of the 10th ACM SIGCOMM Conference on Internet measurement (IMC), 2010;

[71] L. Wang and J. Manner: "*Energy consumption analysis of Wlan, 2g and 3g interfaces",* in Proceedings of the IEEE/ACM International Conference on Green Computing and Communications (GreenCom), International Conference on Cyber, Physical and Social Computing (CPSCom). IEEE Computer Society, 2010;

[72] M. Asplund, A. Thomasson, E. J. Vergara, and S. Nadjm-Tehrani: *"Software-related energy footprint of a wireless broadband module*", in Proceedings of the 9th ACM International Symposium on Mobility Management and Wireless Access, MobiWac '11. ACM, 2011;

[73] V. Konoonen and P. Paakkonen. *"Optimizing power consumption of always-on applications based on timer alignment"*, in the 3rd International Conference on Communication Systems and Networks (COMSNETS), pg. 1-8. IEEE, Jan. 2011;

[74] B. Zhao, Q. Zheng and G. Cao. *"Energy-Aware Web Browsing in 3G Based Smartphones", 2013* IEEE 33rd International Conference on Distributed Computing Systems;

[75] T. Heinz. HiPAC *"High Performance Packet Classification for Netfilter",* phd thesis 28.02.2004;

[76] OMAP Mobile Processors 4 Platform – Texas Instruments. http://www.ti.com/general /docs /wtbu/wtbuproductcontent.tsp?contentId=53247&navigationId=12842&templateId=612;

[77] Battery and Energy Technologies. Information from webpage http://www.mpoweruk.com /performance.html;

[78] GDM-8246 tool, http://www.gwinstek.com/en/product/productdetail.aspx? pid=39&mid =80 & id=198;
[79] Android Play Store for Aplication at https://**play**.google.com/**store**?hl=en;
[80] K. Wehrle, F. P¨ahlke, H. Ritter, D. M¨uller and M. Bechlerm. The Linux Networking Architecture: Design and Implementation of Network Protocols in the Linux Kernel. Prentice Hall, April 2004.
[81] Netfilter dhe librarite, www.netfilter.org/projects/libnetfilter_queue/;
[82] Nokia Energy Profiler, NEP at: http://nokia-energy-profiler.en.softonic.com/symbian/download
[83] http://www.umtsworld.com/technology/overview.htm;
[84] http://www.umts-forum.org/;
[85] Jochen H. Schiller, *"Wireless communications",* 2nd edition London, 2003;
[86] Heikki Kaaranen, *"UMTS networks: architecture, mobility, and services",* Wiley publishing, 2005;
[87] V. Anannd: "*Multi-Interface Connectivity On Modern Mobile",* May 08, 2014. A thesis submitted to the Faculty of the Graduate School of Buffalo, State University of New York;
[88] B. Tiwana, Z. Qian, Z. Wang and R.P. Dick. "*Accurate online power estimation and automatic battery behaviour based power model generation for smartphones*", in Proceedings of the 8th IEEE/ACM/IFIP International Conference on Hardware/software, code sign and system synthesis (CODES/ISSS), 2010;
[89] N. Ding, D. Wagner, X. Chen, A. Pathak, Y. C. Hu, and A. Rice. Characterizing and modeling the impact of wireless signal strength on smartphone battery drain. In Proceedings of the ACM SIGMETRICS/ International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS '13, pg. 29-40. ACM, 2013;
[90] https://developer.android.com/training/index.html ;
[91] R. Kashinsky and H. Balkrishnan: *"Minimizing Energy for Wireless Web Access with Bounded Slowdown",* ACM Wireles Networks, 111 (1-2): pg. 135-148, January 2005;
[92] T. Simunik, L. Benini, P. W. Glynn, and G. DeMicheli: *"Dynamic power management for portable system"*, in MobiCom 2000;
[93] http://www.tldp.org/HOWTO/html_single/TCP-Keepalive-HOWTO/;
[94] R. Fielding, J. Gettys, T. Berners-Lee, et al. RFC 2616: Hypertext Transfer Protocol {HTTP/1.1, Jun 1999. http://www.ietf.org/rfc/rfc2616.txt.,Section 8.1: Persistent Connections;
[95] W. Richard Stevens. TCP/IP Illustrated, Volume 1. Addison-Wesley Professional, 1993. Chapter 23: TCP Keep alive Timer;
[96] http://www.wireshark.org/docs/man-pages/tshark.html;
[97] 3GPP TS3/43.060, General Packet Radio Service (GPRS); Service description; Stage 2;
[98] 3GPP TS3/44.008, Mobile radio interface layer 3 specification, Core Network Protocols-Stage 3;
[99] 3GPP TS 25.401, Technical Specification Group Radio Access Network: UTRAN Overall Description;
[100] 3GPP TS3/45.41, UTRAN Iu interface Layer 1;
[101] 3GPP TS 25.412, UTRAN Iu Interface Signalling Transport;
[102] 3GPP TS 25.413, Technical Specification Group Radio Access Network: UTRAN Iu interface Radio Access Network Application Part (RANAP) signalling;
[103] 3GPP TS3/45.414, UTRAN Iu interface data transport & transport signalling;
[104] 3GPP TS 25.415, Technical Specification Group Radio Access Network: UTRAN Iu interface user plane protocols;
[105] Odin Tools for Android recovery. https://odindownload.com/;
[106] D. Chalmers and M. Sloman:"*A survey of quality of service in mobile computing environments. Communications Surveys and Tutorials*", 2(2):pg. 2-10, IEEE, 1999;
[107] J. Drechsel. Selected topics in cooperative game theory. In Cooperative Lot Sizing Games in Supply Chains, volume 644 of Lecture Notes in Economics and Mathematical Systems, pg. 5-39. Springer, 2010.
[108] M. El-Gendy, A. Bose, and K. Shin. "*Evolution of the internet QoS and support for soft real-time applications",* in the Proceedings of the IEEE, 91(7): pg. 1086-1104, July 2003;

[109] L. Feeney and M. Nilsson. "*Investigating the energy consumption of a wireless network interface in an ad hoc networking environment",* in Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies. INFOCOM 2001, pg. 1548-1557 vol.3, 2001;

[110] **L. Ruci**, L. Karcanaj, O. Shurdi: *"Priority Based Scheduling Algorithm on Smartphones UE",* ISTI 2014, 5th International Conference: "Information Systems and Technology Innovations: projecting trends to a New Economy". www.conference.ijsint.org, Tirana, June 6-7, 2014, Department of Mathematics, Statistics and Applied Informatics, Faculty of Economy, University of Tirana;

[111] L. Guo, X. Ding, H. Wang, Q. Li, S. Chen, and X. Zhang. *"Cooperative relay service in a wireless LAN".* IEEE Journal on Selected Areas in Communications, 25(2): pg. 355-368, February 2007;

[112] T. Hoffeld and A. Binzenhofer. "*Analysis of Skype VoIP traffic in UMTS: End-to-end QoS and QoE measurements."* Computer Networks, 52(3): pg. 650-666, Elsevier. Feb. 2008;

[113] H. Holma and A. Toskala. WCDMA for UMTS: *HSPA Evolution and LTE*. Wiley Online Library: Books. John Wiley & Sons, Ltd., 2010;

[114] C.-Y. Huang, C.-M. Chen, S.-P. Yu, S.-Y. Hsu, and C.-H. Lin. *"Accelerate in-line packet processing using fast queue"*, in Proceedings of IEEE TENCON 2010, Nov 2010;

[115] **L. Ruci**, L. Karcanaj: "Three main pain points from today's Smartphones", (Control Theory and Informatics www.iiste.org, Nov. 2013, Vol. 3, No. 6, pg. 17—24, ISSN 2224-5774 (Paper) ISSN 2225-0492 (Online);

[116] https://en.wikipedia.org/wiki/Netlink;

[117] **L. Ruci,** L. Karcanaj, O. Shurdi: *"Energy Efficiency Combined SW Techniques on Mobiles Android OS"*, SpliTech 2016, 2016 International Multidisciplinary Conference on Computer and Energy Science, Jul. 13-15, 2016, University of Split (FESB). ISBN 978-953-290-063-7, IEEE Catalog Number CFP16F09-ART (Technically co-sponsored by IEEE Communications Society),Split, Croatia;

[118] A. Rice and S. Hay. "Measuring mobile phone energy consumption for 802.11 wireless networking". Pervasive Mobile Computing, 6(6):pg. 593-606, Elsevier. Dec. 2010;

[119] D. Soldani, D. Chiavelli, J. Laiho, M. Li, N. Muhammad, G. Giambiasi, and C. Rodriquez. *"QoE and QoS Monitoring"*, pg. 315-384. John Wiley & Sons, Ltd, 2006;

[120] J. Torsner and J. Bergstrom. *"Direct transition to cell dynamic host configuration"* (DCH), 11 2012;

[121] http://www.lens.org/lens/patent/US_8320929_B2;

[122] https://www.eclipse.org/downloads/packages/eclipse-android-developers;

[123] T. Zhang, Y.-J. Chen, C.-W. Chang, C.-Y. Yang, T.-W. Kuo, and T. Chen. *"Power management strategies in data transmission".* ASPDAC '11, in Proceedings of the 16th Asia and South Pacific Design Automation. pg. 668-675. IEEE, 2011;

[124] **L. Ruci**, L. Karcanaj: *"MDSA, Multi Decision Scheduling Algorithm for UE energy power saving on mobile networks" ,* (European Scientific Journal, October 2013 edition vol.9, No.30, pg. 30-48, ISSN: 1857 – 7881 (Print) e - ISSN 1857- 7431);

[125] http://www.econ.umn.edu/~kimx1423/_content/JMP_MinjungKim.pdf;

[126] Samsung Open Source SW per Samsung S3/4 GT I9100: https://opensource.samsung.com/reception/;

[127] http://sourcery.mentor.com/public/gnu_toolchain/arm-none-linux-gnueabi/;

[128] https://play.google.com/store/apps/;

[129] http://inai.de/documents/Netfilter_Modules.pdf;

[130] http://dl.google.com/android/ndk/android-ndk-r9-linux-x86_64.tar.bz2 /32 bit;

[131] http://dl.google.com/android/ndk/android-ndk-r9-linux-x86_64.tar.bz2 /64 bit;

[132] N. Ding, D. Wagner, X. Chen, A. Pathak, Y. C. Hu, and A. Rice. Characterizing and modeling the impact of wireless signal strength on smartphone battery drain. In Proceedings of the ACM SIGMETRICS/ International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS '13, pg. 29-40. ACM, 2013;

[133] https://tools.ietf.org/rfc/rfc3366.txt;

[134] https://en.wikipedia.org/wiki/Unix_domain_socket;

[135] Porting process on Android, http://elinux.org/Android_Porting;
[136] Netfilter solution examples, https://www.netfilter.org/documentation/HOWTO/netfilter-hacking-HOWTO.html#toc4;
[137] Linkend lists on Linux, https://www.cs.uic.edu/~hnagaraj/articles/linked-list/;
[138] I.Y.Tsai. "Linux Network Architecture outline", *2010/09/02* © by. Outline , http://iaic.csie.tku.edu.tw/netintro/Linux/20100902-LinuxNetworkArchitectureS.pdf;
[139] Linux buffering, https://wiki.linuxfoundation.org/networking/sk_buff;
[140] MP3 songs file of 6.18 MB size under http://www.KerkojMuzik.com;
[141] Google Chrome Speed Tests video files 19.2MB, http://www.youtube.com /watch ?v=n CgQDj iotG0;
[142] *SystemSens:* monitoring usage in smartphone research deployment at, http://falaki.net/pub/ **SystemSens**.pdf;
[143] Nokia Lumia 625 HW and SW *specification, http://www.nokia.com/global/products/phone/lumia625/;*
[144] Little Eye, Performance Analysis and Monitoring tool. *http://www. littleeye.co/;*
[145] ARM 2010q1 toolchain. *https://sourcery.mentor.com/sgpp/lite/arm/portal/release;*
[146] *www.roman10.net/2011/07/23/a-linux-firewall-using-netfilter-part-1overview/*
[147] http://www.roman10.net/2011/07/24/linux-kernel-module-basics-and-hello-world/
[148] ADB Android. https://developer.android.com/studio/command-line/adb.html
[149] Android Studio. https://developer.android.com/studio/intro/index.html

Botime e Konferenca

L. Ruci, L.Karcanaj, "*Evoluimi i teknologjisë mobile dhe i shërbimeve të internetit, zgjidhja e problematikave*", Tiranë, Shqipëri, Korrik 06-07, 2012 në rëvistën "Hulumtime Shkencore", UKT, me ISSN 2226-1605.

L. Ruci[1], L.Karcanaj,O.Shurdi, "*Smartphones' Influence and challenges for end users and Mobile operators*", www.ijsint.org, Vol 1 No. 7, July 2013 (Internacional Journal of Science Innovation new Technologies) pg 55—60, ISSN:2223-2257 (Print), 2225 – 0751 (online).

L. Ruci[1], L.Karcanaj, "*MDSA, MULTI DECISION SCHEDULING ALGORITHM FOR UE ENERGY POWER SAVING ON MOBILE NETWORKS*", (European Scientific Journal, October 2013 edition vol.9, No.30, pg. 30 - 48, ISSN: 1857 – 7881 (Print) e - ISSN 1857- 7431).

L. Ruci[1], L.Karcanaj, "*Three main pain points from today's Smartphones*", (Control Theory and Informatics www.iiste.org, November 2013, Vol.3, No.6, pg. 17 - 24, ISSN 2224-5774 (Paper) ISSN 2225-0492 (Online).

L. Ruci[1], L.Karcanaj,O.Shurdi : "Priority *Based Scheduling Algorithm on Smartphones UE* " , ISTI 2014, 5th International Conference: "Information Systems and Technology Innovations: projecting trends to a New Economy". www.conference.ijsint.org, June 6-7, 2014, DEPARTMENT OF MATHEMATICS, STATISTICS AND APPLIED INFORMATICS, FACULTY OF ECONOMY, UT, Tirana, Albania.

O.Shurdi[1], R.Miho, **L.Ruci**: "*Analyzing VoIP over WLAN, 3G, WIMAX and LTE issues* " , ISTI 2014, 5th International Conference: "Information Systems and Technology Innovations: projecting trends to a New Economy". www.conference.ijsint.org, Tirana , June 6-7, 2014, DEPARTMENT OF MATHEMATICS, STATISTICS AND APPLIED INFORMATICS, FACULTY OF ECONOMY, UT, Tirana, Albania.

L. Ruci[1], L.Karcanaj, O.Shurdi: *Nokia Windows Mobile's Power Consumption Measurements and Analysis "* pg. 1-12, ICT Innovations 2014, Web Proceedings ISSN 1857-7288 © ICT ACT, 2014, 6th ICT Innovations Conference, (ICT ACT http://ict-act.org, Skopje 2014, On line edition published on http://ictinnovations.org) 9-12 September 2014, Ohrid, R.Macedonia,

L. Ruci[1], L.Karcanaj,O.Shurdi : "*Energy Efficiency Combined SW Techniques on Mobiles Android OS* " , SpliTech 2016, 2016 International Multidisciplinary Conference on Computer and Energy Science, July 13-15, 2016**,** University of Split (FESB). ISBN 978-953-290-063-7, IEEE Catalog Number CFP16F09-ART (Technically co-sponsored by IEEECommunications Society). Split, Croatia.

O.Shurdi, L.Ruçi, V.Kolici, A.Lala, and B.Kamo: Android OS Stack Power Management Capabilities, pg 754-765, The 6th International Conference on Emerging Internet, Data & Web Technologies (EIDWT-2018), March 15-18 , 2018, Tirana, Albania