VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

PRAKTIKOS ATASKAITA

Praktiką atliko: **Tautvydas Aleknavičius (+37064021194)**
(studento vardas, pavardė) (parašas)

Informatika, 4 kursas
(studijų programa, kursas)

Praktikos institucija: UAB “8devices”
(organizacijos pavadinimas)

Organizacijos praktikos vadovas VP of Embedded devices Žilvinas Valinskas
(pareigos, vardas, pavardė)

Organizacijos praktikos vadovo įvertinimas: _____
(įvertinimas, parašas)

Universiteto praktikos vadovas: _____
(mokslo laipsnis, vardas, pavardė)

_____
(parašas)

Ataskaitos įteikimo data _____

Registracijos Nr. _____

Įvertinimas _____
(data, įvertinimas, parašas)

Vilnius, 2021

Turinys

Įvadas	**3**
1. Praktikos vietos aprašymas	**4**
2. Darbo sąlygos	**5**
3.Projektai	**6**
3.1 Virtualios realybės akinių siųstuvai	**6**
3.2 LTE modulio integracija	**7**
4. Projektų detali apžvalga	**8**
4.1 JavaScript testų automatizavimo procesas bei integracija.	**8**
4.2 "Wi-Fi" Multimedijos sustiprinta paskirstyto kanalo prieiga	**10**
4.3 Laiko sinchronizavimas tarp kompiuterių Linux aplinkoje	**12**
4.4 Darbas su OpenWRT operacine sistema	**14**
5. Apibendrinimas	**15**
5.1 Atlikti darbai	**15**
5.2 Išvados	**16**

Įvadas

Praktiką atlikau įmonėje UAB “8devices”, kurioje dirbu jau ne pirmi metai. Ši įmonė specializuojasi Wi-Fi produktų gamyboje nuo įrenginio prototipo sukūrimo iki programinės įrangos pritaikymo bei palaikymo. Įmonė aptarnauja bei teikia paslaugas daugeliui programinės bei aparatūrinės įrangos gamintojų pasaulyje.

Praktikos metu dirbau prie keleto projektų, kurie kiekvienas buvo skirtingi ir įdomūs. Turėjau galimybę ir vis dar dirbu prie 60 GHz projekto su kanadiečių įmone “Peraso Technologies”, kuri gamina 60 GHz radijas bei teikia programinės įrangos paslaugas. Su šia įmone dirbau prie vieno projekto. Tai yra virtualios realybės akiniams skirtų siųstuvų testavimas bei automatizuotų testų rašymas. Šiame projekte, daugiausia dirbau rašydamas automatinius testus JavaScript programavimo kalba. Kitas projektas yra susijęs su LTE modulio integracija į “8devices” įmonės gaminamus įrenginius ir pateikimas klientui. Šiame projekte turėjau galimybę ir įvykdyti kliento reikalavimus ir juos ištestuoti. Atliekant praktiką reikėjo nagrinėti bei išmokti kompiuterinius tinklus (angl. networking), susipažinti su įvairiais Wi-Fi (bevielio ryšio) standartais, OpenWRT sistema. Darbas visą laiką vyko Linux operacinės sistemos aplinkoje. Darbo metu daugiausia reikėjo skaityti įvairių Wi-Fi standartų dokumentacijas. Taip pat buvo gera proga susipažinti su Perforce versijavimo sistema (prieš tai buvau naudojęs tik GIT), Jira projekto valdymo bei problemų sekimo sistema, taip pat komponentų kompiliavimo bei integracijos sistema TeamCity. Susipažinimas su šiais įrankiais išties bus naudingas ateityje.

Praktikos metu buvo naudojami įvairūs informacijos šaltiniai. Dažniausiai tai buvo standarto dokumentacijos arba vidiniai įmonės resursai Confluence aplinkoje, tačiau jeigu reikėtų išskirti šaltinius, norėčiau paminėti šiuos:

- https://openwrt.org/
    - https://openwrt.org/docs/guide-user/network/wan/wwan/ltedongle
    - Dar daug kitų puslapių susijusių su OpenWRT kompiliavimu
- https://linux.die.net/
    - https://linux.die.net/man/8/ptp4l
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference

Praktikos metu pasiekti rezultatai, leido mūsų komandai išleisti kuo kokybiškesnę ir gerai ištestuotą programinę įrangą, kurią vėliau naudojo bei naudoja mūsų klientai.

Automatiniai testai surado ne vieną problemą, kuri ne vieną kartą atidėjo programinės įrangos išleidimą klientams bent savaitei, kol problema buvo sutvarkyta.

1. Praktikos vietos aprašymas

“8devices” įmonė - viena iš nedaugelio Lietuvos įmonių, teikianti klientams paslaugas, kurios apima visus etapus, nuo projekto pradžios (planavimo), aparatūrinės įrangos pagaminimo iki pilno veikiančio įrenginio su programine įranga. Įmonėje yra keletas komandų, kurių kiekviena dirba su skirtingais projektais. Tai yra aparatūrinės įrangos (hardware) gaminimo, testavimo komandos, programinės įrangos kūrimo, testavimo komandos, kurių yra net keletas. Įmonės pagaminti Wi-Fi produktai dažniausiai yra integruojami robotikos srityje arba į kitus įrenginius, suteikiant bevielę prieigą. Anksčiau įmonė buvo žinoma, dėl darbo, su tokiomis įmonėmis, kaip LigoWave, IgniteNet. Įmonė specializuojasi maršrutizatorių gamyboje (angl. embedded devices) bei jų programinės įrangos kūrimu. Šiuo metu yra daug daugiau projektų, vykstančių vienu metu. Įmonė stipriai plečiasi ir yra orientuota į užsienio rinką. Kiekvieną metų ketvirtį atsiranda bent po vieną naują projektą ar klientą.

Įmonė yra įsikūrusi naujai atidarytame “Vilnius Tech Park”. Kolektyvas yra labai draugiškas ir visada pasiruošęs padėti. Komandos suskirstytos į kabinetus po 5-8 žmones, todėl dirbti patogu, nes nebūna daug triukšmo. Virtuvėje visada yra vaisių bei programuotojų pagrindinio kuro - kavos. Beveik visą laiką dirbau iš namų dėl karantino, todėl daug teko bendrauti nuotoliniu būdu. Dėl nuotolinio darbo nepatyriau jokių nesklandumų, nes įmonė naudoja VPN pasiekti vidiniams resursams, o bendravimui Teams arba RocketChat.



1 pav. “8devices” logotipas

2. Darbo sąlygos

Įmonė aprūpina visais reikalingais įrankiais bei aplinka darbui atlikti. Kiekvienas darbuotojas gali išsirinkti kompiuterį, kuris dažniausiai būna nešiojamas. Taip pat esant reikiamybei, galima užsisakyti klaviatūrą, pelę ir papildomą įrangą. Niekada nebuvo jokių problemų su įrangos neturėjimu. Jeigu kompiuteris sugenda, jis yra pakeičiamas nauju.

Darbo grafikas yra lankstus. Einant į ofisą, galima dirbti savo pasirinktu laiku. Kai kurie žmonės pradeda darbą nuo 6 ryto ir dirba 8 valandas. Dažniausiai darbas yra pradedamas apie 9 valandą ir baigiamas apie 18 valandą. Dirbant su partneriais ir už Atlanto, darbo laikas gali keistis, nes yra didelis laiko skirtumas, tačiau tai priklauso labiau nuo pačio darbuotojo noro.

Įmonės kolektyvas sudarytas iš įvairaus amžiaus žmonių. Gali rasti ir tokio pačio amžiaus žmonių, su bendrais pomegiais, todėl visada bus apie ką pabendrauti. Įmonėje dirba daug labai kompetentingų ir daug išmanančių žmonių, todėl kiekvieną dieną išmoksti kažką naujo.

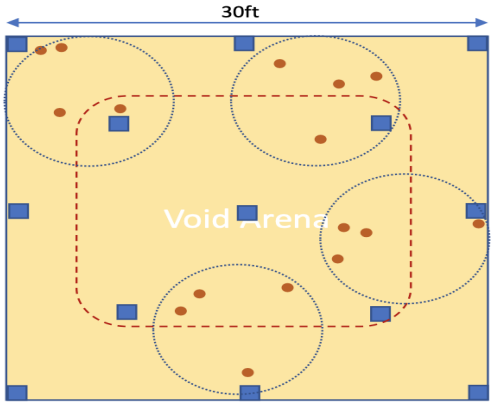
Prieš karantiną įmonė rengė įvairius renginius (kalėdinius bei vasaros), kurių metu darbuotojai geriau susipažįsta vieni su kitais bei gerai praleidžia laiką. Taip pat įmonei priklauso ir lauko terasa, kurioje vasarą galima pietauti arba tiesiog dirbti, leisti laiką. Pietų laikas įmonėje nėra nustatytas. Pietauti galima pačioje įmonėje arba pasirinktoje vietoje.

Darbo aplinką bei galimybes įmonėje vertinu labai gerai. Dirbant šioje įmonėje, darbas niekada nebuvo nuobodus, nes visada atsiranda naujų iššūkių.

3.Projektai

3.1 Virtualios realybės akinių siųstuvai

Didžiąją dalį laiko, dirbau prie projekto su mūsų partneriais iš Kanados, įmone “Peraso Technologies”. Ši įmonė specializuojasi 60GHz didelio pralaidumo radijų bei jų programinės įrangos gamyba bei integracija. Projekto tikslas buvo padaryti virtualios realybės kambarį, kuriame daugelis žmonių vienu metu galėtų žaisti žaidimus ir gauti aukštos kokybės vaizdą realiu metu. Projekto vizija buvo turėti kambarį, kuriame yra sudėlioti siųstuvai (angl. Access points), kurie padengia visą kambario plotą. Visus siųstuvus reikėjo valdyti, todėl jie visi buvo prijungti prie valdiklio (angl. controller), kuris srautus skirstė po siųstuvus ir darė kitus sprendimus, tokius, kaip, kad spręsti, kada prijungti imtuvą prie kito siųstuvo.



2 pav. Žaidimų kambario padengimo vaizdas

Ant kiekvieno žmogaus akinių buvo pritvirtinti imtuvai, kurie gaudavo žaidimo vaizdą realiu laiku. Žmonės galejo judėti po kambarius neprarasdami jokių duomenų ir matydami tą patį vaizdą. Sunkiausias šio projekto etapas yra padaryti, kad kiekvienas imtuvas būtų sinchronizuotas bei visų jų paklaida būtų ne didesnė nei 2 mikrosekundės. Žmonėms judant ir jungiantis nuo vieno siųstuvo prie kito, labai svarbus aspektas yra tai, kad žmogus nepajaustų persijungimo ir gautų visus duomenų paketus, kuriuos turėjo gauti. Šiame projekte gavau galimybę dirbti prie kelių testavimo planų. Rašant ir leidžiant testavimo planus, didžiausia problema buvo įsitikinti, kad prieš testą, visi siųstuvai yra teisingai sinchronizuoti. Jeigu siųstuvai bus nesinchronizuoti, tuomet imtuvai turės dar didesnę paklaidą, kuri galiausiai taps matoma plika akimi vartotojui. Kai paklaida tokia maža, testai labai dažnai nebūdavo sėkmingi. Prie šio projekto iš viso dirbo 6 programinės įrangos testuotojai, kurie visi buvo kanadiečiai, išskyrus mane. Aš gavau galimybę dirbti prie srautų prioritizacijos bei

laiko sinchronizavimo projektų. Kiekvieno šių užduočių detalesni aprašymai pateikiami vėliau. Projekto metu labiausiai reikėjo pritaikyti JavaScript programavimo kalbos bei kompiuterinių tinklų žinias. Taip pat nemažai reikėjo domėtis kompiuterių sinchronizavimo įrankiais Linux aplinkoje. Dokumentacijos skaitymas yra kitas labai svarbus dalykas, kiekviename. Prieš kiekvieną užduotį labai svarbu turėti reikalavimus bei dokumentaciją, kad galėtum teisingai suplanuoti darbus. Projektas susidėjo iš daugelio etapų, kurių pirmas buvo įsitikinti, jog mūsų programinė įranga gali valdyti ir skirstyti srautą, kuris yra skirtingo svarbumo.

3.2 LTE modulio integracija

Šio projekto apimtis nėra tokia didelė kaip prieš tai minėto projekto. Prie jo dirbau tik aš ir dar keletas žmonių iš aparatūrinės įrangos gamybos bei testavimo komandos. Pagrindinis šio projekto tikslas buvo esamam produktui padaryti pakeitimus techniškai (angl. hardware) bei programiškai (angl. software). Mano dalis buvo pritaikyti įrenginį iš programinės dalies pusės.

Taigi, originaliai mūsų gaminamas įrenginys turėjo 5 interneto kabelio jungtis, reikėjo padaryti, jog įrenginys turėtų tik 4 jungtis, bei įsitikinti, kad trys iš jų yra skirti vietiniam tinklui (LAN), o vienas skirtas globaliam tinklui (WAN) pajungti. Taip pat reikėjo integruoti LTE modulį, bei jo konfigūravimo sistemą. Galiausiai reikėjo pasirūpinti, kad kiekviena lemputė esanti ant įrenginio, rodytų tai, ką klientas nori.

Projekto metu labiausiai susipažinau su OpenWRT programine įranga, kuri yra paremta GNU/ Linux ir yra naudojama daugelyje maršrutizatorių. Išmokau ne tik susikompiliuoti programinę įrangą savo įrenginiui, tačiau ir įdiegti/išdiegti įvairias tvarkykles, integruoti naują funkcionalumą, keisti programinį kodą (angl. source code) pagal tai, ko nori klientas bei konfigūruoti įrenginį. Didžiausia problema, su kuria susidūrėme darydami šį projektą yra LTE modulio veikimo testavimas. Susidūrėme su situacija, kai išorinis modulis veikia tik ~50% kartų po perkrovimo.

4. Projektų detali apžvalga

Šiame skyriuje aptarsiu kiekviename projekte naudotus įrankius bei technologijas. Žinių tobulinimas šiose srityse padėjo pasiekti praktikos tikslus bei sėkmingai įgyvendinti projektus bei užduotis.

Pirmąjame projekte pagrindinė užduotis bei domėjimosi šaltiniai buvo daugiausiai susiję su JavaScript, 802.11ad Wi-Fi standartu bei laiko sinchronizacija. Taip pat buvo galimybė suprasti, kaip vyksta automatizavimo procesas, kokie yra jo etapai.

Antrajame projekte pagrindinė užduotis buvo gilinimasis į OpenWRT operacinės sistemos kompiliavimo, integravimo bei konfiguracijos subtilybes.

4.1 JavaScript testų automatizavimo procesas bei integracija.

Testų automatizavimo procesas susideda iš daugelio etapų, kurie yra vienodai svarbūs, kaip ir pati implementacija. Pirmasis etapas yra produkto reikalavimų (angl. Product requirement) supratimas ir analizavimas. Tam kad parašytum testų rinkinį specifiniam funkcionalumui ištestuoti, turi turėti sąrašą dalykų, kurie yra reikalaujami iš funkcionalumo. Taigi, pradžioje reikia tiesiog skaityti dokumentacijas. Po šio proceso vyksta sekantis procesas, “pradinis” susirinkimas (angl. kick off meeting), kuriame susirenka projekto vadovas, programinės įrangos kūrėjai bei testuotojai. Šio susirinkimo metu visi aptaria, ar reikalavimai buvo suprasti teisingai bei aptaria detales.

Sekantis procesas yra abstraktus testo dokumentacijos rašymas. Šis procesas yra skirtas jau atskirų testų rašymui, jų nedetalizuojant. Reikia suskaidyti testavimo planą į kuo daugiau skirtingų testų rinkinių (angl. test suites) bei pridėti kuo daugiau testų su abstrakčiais aprašymais (angl. outlines) į šiuos rinkinius. Testų pavyzdžiai galėtų būti neigiami (angl. negative) testai, kuriuose darome neleidžiamus, nevalidžius veiksmus ir tikimės jog sistema veiks nepriekaištingai. Taip pat yra teigiami (angl. positive), našumo (angl. performance), apkrovos (angl. stress), bei realaus scenarijaus (angl. Use case) testai. Šis etapas yra svarbus, nes pradeda ryškėti testų kontūrai. Šio etapo pabaigoje, vėl įvyksta minėtų komandų susirinkimas aptarti testavimo planą, pasakyti pastabas arba pridėti kažką į testą. Taip pat šiame etape yra apibrėžiamas testų prioritetas ir nurodoma apytikslė data, testams parašyti.

Trečiasis etapas yra atskirų testų detalizavimas bei aprašymas. Kiekvienas testas turi tam tikras charakteristikas, tokias kaip - testo aprašymas, kurį mes jau atlikom praeitame etape, testo išankstinės sąlygos (angl. pre-conditions), reikalavimai, topologija,

priklausomybes (angl. dependencies), testo įvestis bei procedūra. Testo įvestis yra naudojama nustatyti, kokių parametrų testas turėtų tikėtis. Šiuo atveju tai gali būti srauto generatoriaus parametrai (duomenų srauto laikas, kiekis, tipas bei laukiamas rezultatas). Testo procedūra yra bene svarbiausia testo dalis. Šioje dalyje reikia aprašyti kiekviena testo žingsnį, ka tuo žingsniu tikimės pasiekti bei ką darome tam žingsniui nepavykus. Šio etapo pabaigoje vyksta dar vienas minėtų komandų susirinkimas. Susirinkimo metu aptariami detalūs testų žingsniai ir įsitikinama, kad testų dokumentacijose nėra klaidų. Tai yra paskutinis žingsnis prieš testų implementaciją.

Ketvirtasis etapas yra testų implementacija JavaScript programavimo kalba. Šie testai yra tiesiog prieš tai patvirtintų žingsnių vykdymas naudojant JavaScript kodą. JavaScript testais mes konfigūruojame radijas, sujungiame jas, leidžiame interneto srautą iš vienos radijos į kitą bei darome kitus veiksmus. Testai turi kelias bibliotekas. Viena iš jų yra pavadinimu “PvUtils”, kurioje mes rašome didžiąją dalį logikos, kurią vėliau naudojame savo testuose. Kita dalis yra “PvSteps”, kurioje mes aprašome tam tikrų žingsnius ir ko iš jų tikimės. Galiausiai, visi testai yra suskirstyti į atskirus rinkinius pagal reikalavimus. Taip pat yra dalis, vadinama “Automation” kuri tiesiogiai bendrauja su įrenginiais. Ši dalis komunikuoja su Linux, Windows bei OpenWRT operacinėmis sistemomis, kurios yra mūsų testavimo aplinkose. Praktikos metu turėjau galimybę rašyti tiek JavaScript testus, tiek prisidėti prie “PvUtilities” biblietekos praplėtimo, tiek susipažinti su “Automation” specifika.

```
function PR773_AvatarAPSynchronization_TS5_StressTests() {
    this.tests = [
        PR773_TS5_TC1_TimeSyncWhilePollingMIB,
        PR773_TS5_TC2_TimeSyncPollingMIBandTraffic,
        PR773_TS5_TC3_FirmwareTimeSyncDuringHardReset,
        PR773_TS5_TC4_FirmwareTimeSyncThroughputCycles
    ];
}

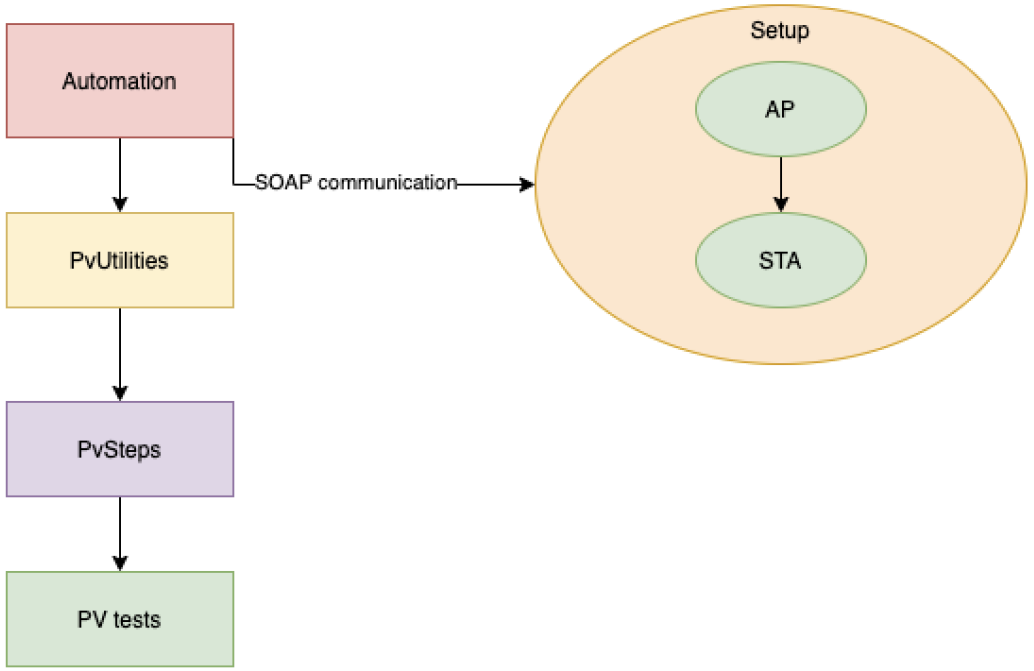
PR773_AvatarAPSynchronization_TS5_StressTests.prototype.PreConfig = function (config, clients) {
    SDSys.PrintLine("------>start TS5 preconfig");
    Pv.General.UniversalCleanup(clients[0].GetAPCPeerList().concat(clients[0].GetSTAPeerList()));
    SDSys.PrintLine("preconfig end");
};


PR773_AvatarAPSynchronization_TS5_StressTests.prototype.PostConfig = function (config, clients) {
    SDSys.PrintLine("------>start TS5 postconfig");
    Pv.General.UniversalCleanup(clients[0].GetAPCPeerList().concat(clients[0].GetSTAPeerList()));
};

exports = PR773_AvatarAPSynchronization_TS5_StressTests;
```

3 pav. Penktasis testų rinkinys skirtas testuoti sinchronizacijai esant didelei apkrovai

Galutinis etapas, po testų implementacijos yra TeamCity integracija. Visi testai paleidžiami per TeamCity sistemą. Testai yra leidžiami arba kompiliuojami panašiai kaip programinė įranga. Testas yra priklausomas nuo daugybės kitų komponentų. Leidžiant testą susidaro visa eilė komponentų, kurie turi būti sukompiliuoti prieš testą ir nusiųsti į testo vykdymo direktoriją. Jeigu neatsitiko jokių nenumatytų situacijų yra grąžinamas 0. Jeigu testo metu kažkas nepasisekė, grąžinamas klaidos kodas. Visais atvejais testo gale yra pateikiama visų testų glausta statistika. Taigi parašius testus, reikia parašyti tam tikras taisykles, kad testai atsirastų TeamCity aplinkoje ir būtų prieinami visiems. Dažniausiai taisyklės būna JSON formatu.

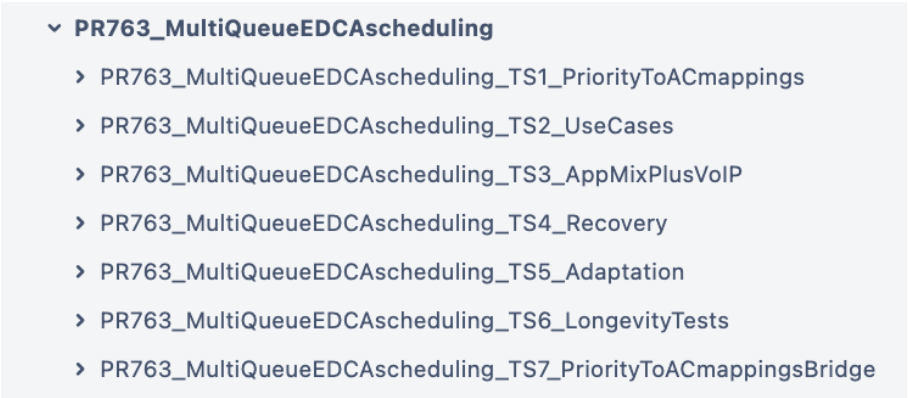


4.2 "Wi-Fi" Multimedijos sustiprinta paskirstyto kanalo prieiga

Kaip jau minėta anksčiau, tam kad įvykdyti virtualios realybės siųstuvų projektą, pirmiausia reikėjo pasirūpinti, jog radijos tvarkyklė (angl. driver) ir programinė įranga (angl. firmware) gali skirstyti srautus pagal prioritetus. Kadangi virtualios realybės akiniuose daugiausia srauto užima vaizdas bei garsas, reikėjo įsitikinti, jog šie du srauto tipai gaus didesnį prioritetą duomenų eilėje nei kiti duomenų paketai. Siunčiant srautą iš vienos radijos į kitą, srautas turi pasiskirstyti pagal tam tikras taisykles. Tai yra vadinama "Wi-Fi"

multimedijos sustiprinta paskirstyto kanalo prieiga (angl. Wi-Fi multimedia enhanced distributed channel access).

Pradedant nuo pradžių, kiekvienas duomenų paketas kompiuterių tinkluose gali turėti tam tikrą reikšmę, kuri nusako jo prioritetą nuo 0 iki 224. Aukščiausia reikšmė reiškia, jog prioritetas yra didžiausias. Šis skaičių intervalas yra padalintas į 8 rėžius, kurie ir nusako prioritetą. Taigi žemiausias prioritetas yra vadinamas “geriausias bandymas” (angl. Best effort). Šis srautas būna pažymėtas 0 arba 3 prioriteto lygiu, todėl tam tikrais atvejais gali būti svarbesnis nei sekantis prioritetas. Šiam srautui yra priskiriamas naršymas internete, įvairių puslapių atidarinėjimas, kuris nereikalauja staigaus užkrovimo. Tai taip pat gali būti elektroninio pašto laiškų siuntimas bei atidarymas. Šis srautas nėra jautrus paslaugų kokybės metrikoms, tokioms kaip paketų praradimas, vėlinimas arba srauto netolygumui (angl. Jitter). Aukštesnis prioritetas yra foninio srauto (angl. background). Šis srautas gali turėti prioritetų lygius 1 ir 2. Tai gali būti vaizdo srautas, kuris pradžioje yra buferizuojamas ir užkraunamas ir tik vėliau rodomas. Pavyzdys yra Youtube, Netflix. Trečiasis prioritetas, kuris atitinka 4-5 prioritetų lygį yra vaizdas (angl. video). Šis srautas yra dažniausiai naudojamas video konferencijų, TV transliacijų metu, kitaip sakant visur, kur vaizdas nėra buferizuojamas, o yra pateikiamas gyvai. Aukščiausias prioritetas yra garso (angl. voice). Šis srautas neužima daug eterio laiko, tačiau bet kokiu atveju gaus daugiausia eterio. Kaip pavyzdį galima pateikti situaciją, kai nuotolinių paskaitų metu pradeda trūkinėti interneto ryšys, pradžioje trūkinėja vaizdas, o tik vėliau sutrinka garsas. Taip yra dėl aukštesnio garso prioriteto ir kitų veiksnių. Taip pat yra ir dar aukštesnis prioritetas, tačiau šiuo atveju jis buvo nebeaktualus. Tai yra tinklų nukreipimo (angl. Routing) prioritetas.



4 pav. Visi testų rinkiniai, esantys EDCA testavimo plane

Taigi, mano testų esmė buvo įsitikinti, jog leidžiant skirtingų prioritetų interneto srautus, visų srautų greičiais pasiskirsto pagal iš anksto nustatytus kriterijus. Apačioje esančioje lentelėje galima pamatyti keletą srautų pasiskirstymo pavyzdžių. Iš viso jų buvo 15.

Vieni testai buvo paprasti, kurie tiesiog paleisdavo interneto srautą, pamatuodavo jį tam tikrą laiką, po kiek laiko paleisdavo antrą srautą, trečią ir taip toliau. Kiti testai buvo sudėtingesni, jų metu reikėjo paleisti tam tikrus srautus ir atlikti srauto bei ryšio atsigavimo scenarijų. Tai yra nutraukti ryšį imituojant tam tikrus ryšio sutrikimus, tiek techninius, tiek programinius. Interneto srautui generuoti naudojome “iperf3” srauto generatorių. Vėlesni testai buvo dar sudėtingesni. Jų metu mes imituodavom tikrą srautą naudodami “ixChariot” srauto generatorių. Šiuo atveju buvo galima pamatuoti, kaip srautas pasiskirsto naudojantis Facebook, naršant naujienų portaluose, kalbant Skype bei žiūrint filmus internetu. Šiems testams buvo naudojami srautai (duomenų paketai), kurie yra identiški realiems scenarijams, ko nebuvo galima padaryti prieš tai naudojant “iperf3” srauto generatorių, nes jis generuoja atsitiktinius paketus, bet uždeda tam tikrą prioritetą. Viso buvo parašyta apie 80 testų bei surasta ne viena problema.

Ataskaitos metu neįmanoma perteikti visos gautos patirties darant tokį darbą, nes labai nesunku nukrypti į detales, kurios nebus labai suprantamos skaitytojui bei intuityvios.

BE/BK/VI/VO	Geriausias bandymas(%)	Foninis srautas (%)	Vaizdas (%)	Garsas (%)
Y /Y /Y/ Y	10	20	30	40
N /Y /Y /Y	n/a	30	30	50
Y /Y /N /N	40	60	n/a	n/a
N /N /Y /Y	n/a	n/a	40	60

1 lentelė. Duomenų srauto pasiskirstymas leidžiant skirtingų prioritetų srautus

4.3 Laiko sinchronizavimas tarp kompiuterių Linux aplinkoje

Įsitikinus, jog programinė įranga veikia tinkamai su skirtingo prioriteto interneto srautais, sekantis žingsnis buvo pasirūpinti, jog visi kompiuteriai testuojamoje aplinkoje yra sinchronizuoti. Mano tikslas buvo pirmiausia išsiaiškinti kaip veikia linux sinchronizavimo įrankis “ptp4l”, integruoti šį irankį į automatinio testavimo aplinką bei naudoti automatiniuose testuose testuose. Mes naudojome du įrankius, kurie priklauso “ptp4l”. Pirmasis yra pats “ptp4l”, kuris naudojamas sinchronizuoti du ar daugiau kompiuterių interneto plokščių, kitas “phc2sys”, kuris naudojamas sinchronizuoti kompiuterio plokštės laiką su kompiuterio pagrindiniu sistemos laiku.

Taigi, pradžioje darėme daug eksperimentų, aiškinomės ko reikia, kad kompiuteriai galėtų būti sinchronizuoti. Sinchronizacijai reikia vieno kompiuterio kuris bus pagrindinis mazgas (angl. Master node) ir vieno ar daugiau kompiuterių, kurie yra dukteriniai mazgai (angl. slave nodes). Visų šių kompiuterių plokštės privalo turėti aparatūrinės įrangos laiko žymėjimą (angl. Hardware timestamping), kuris yra daug tikslesnis nei programinės įrangos laiko žymėjimas. Mūsų topologijoje mes turėjome viena kompiuterį kuris yra kontroliuojantis ir naudoja Linux operacinę sistemą, vieną ar daugiau kompiuterių, kurie turi prijungtus siųstuvus bei naudoja OpenWRT programinę įrangą bei vieną ar daugiau kompiuterių kurie naudojo Windows operacinę sistemą ir turėjo prijungtus imtuvus. Galiausiai išsiaiškinome kokius teisingus parametrus reikia naudoti, norint teisingai sinchronizuoti kompiuterius siųstuvo pusėje. Pasiekėme gana tikslią sinchronizaciją su mažiau nei 100 nano sekundžių paklaida. Apačioje yra pavyzdys, kaip atrodo sinchronizacijos procesas. Laikas svyruoja į abi puses, tačiau jeigu didžiausia paklaida yra 127 nano sekundės, tai dar yra pakankamai geras rezultatas.

```
ptp4l[170.425]: master offset         16 s2 freq  +79233 path delay      3625
ptp4l[171.426]: master offset        -41 s2 freq  +79180 path delay      3646
ptp4l[172.426]: master offset        127 s2 freq  +79336 path delay      3646
ptp4l[173.426]: master offset        -39 s2 freq  +79208 path delay      3656
ptp4l[174.426]: master offset        -19 s2 freq  +79217 path delay      3656
ptp4l[175.426]: master offset        -20 s2 freq  +79210 path delay      3657
ptp4l[176.426]: master offset        -64 s2 freq  +79160 path delay      3657
ptp4l[177.426]: master offset        112 s2 freq  +79317 path delay      3657
ptp4l[178.426]: master offset        -64 s2 freq  +79174 path delay      3657
ptp4l[179.426]: master offset         89 s2 freq  +79308 path delay      3656
```

Sekantis etapas buvo šių dviejų įrankių integracija į testavimo aplinką. Prieš kiekvieną testą mūsų tikslas buvo paleisti visus reikalingus įrankius, kurie sinchronizuodavo reikalingus kompiuterius. Paleidus šias komandas automatiškai, mes kiekvieną sekundę tikriname komandos išvedama informaciją ir jeigu 10 iš eilės sekundžių paklaida neviršija nustatytos ribos, galime gražinti teigiamą rezultatą ir pereiti į sekantį žingsnį. Didžiausia problema buvo išlaikyti sinchronizaciją, kai kompiuterio procesorius yra apkrautas bei tais atvejais, kai vienas iš kompiuterių persikrauna. Padidėjus kompiuterio apkrovai, pasikeičia procesoriaus darbinis dažnis, o nuo jo labai priklauso aparatūrinė (angl. hardware) sinchronizacija.

```
AvatarUtils.general.getApSynchronizationStatus = function (apcObject :Object , avatarApArray :*[] , maxDeltaTolerance){
    var apSyncMessage = "AP synchronization status in "+AvatarUtils.general.getAvatarBssString(apcObject, avatarApArray)+":\n";
    var maxDeltaTolerance = maxDeltaTolerance!=undefined ? maxDeltaTolerance : 500000;
    var resObj = AvatarUtils.general.apSynced(avatarApArray, maxDeltaTolerance);
    return apSyncMessage+JSON.stringify(resObj);
}
```

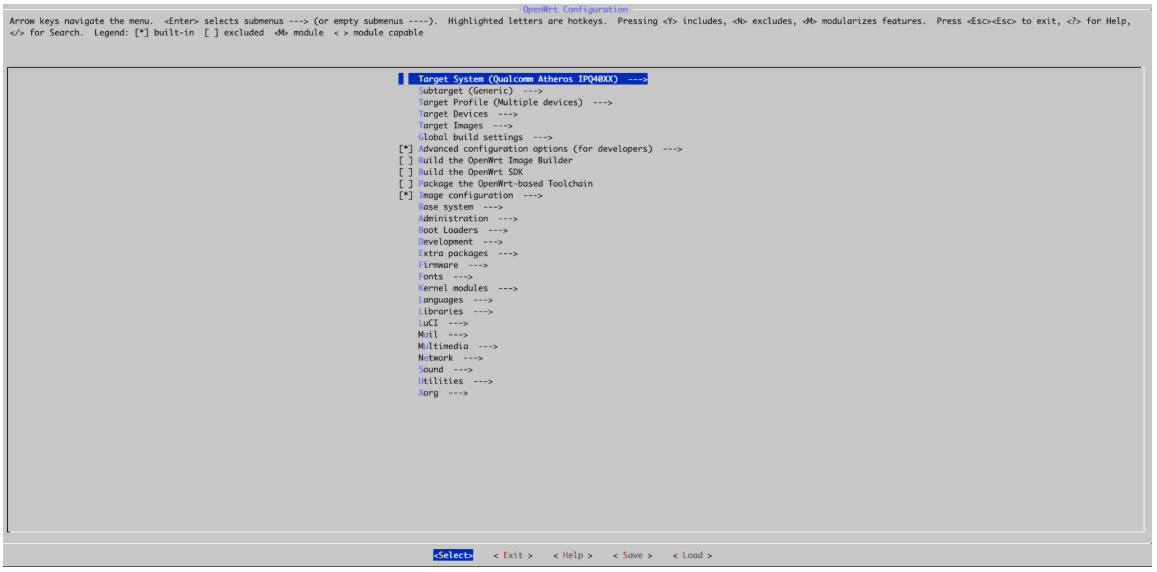
5 pav. Funkcija grąžinanti sinchronizacijos statusą einamuoju momentu.

Galutinis žingsnis buvo sukurti testų rinkinį, kurie tikrino sinchronizaciją ne tik ant kompiuterių, kur yra paleistas “ptp4l” įrankis, bet ir ant radijų, kurios buvo sinchronizuojamos kitaip, siunčiant duomenų paketus belaidžiu ryšiu. Buvo sukurta keli testų rinkiniai, kurie tikrino sinchronizaciją nuo pačių paprasčiausių scenarijų iki sudėtingiausių, kur mes apkrauname procesorių ir tarp radijų sklinda didelis srautas duomenų. Tai yra patys svarbiausi testai visame projekte, nes jeigu sinchronizacija neveikia, vėliau tai persiduoda ir į kitus programinius komponentus ir juos įtakoja.

4.4 Darbas su OpenWRT operacine sistema

Galima sakyti, kad darbas su OpenWRT programine įranga yra pagrindinė užduotis šioje įmonėje, nes dauguma gaminamų produktų viduje turi įrašytą šia programinę įrangą. Tai yra ganėtinai paplitusi sistema tarp interneto maršrutizatorių ir naudojama plačiai visame pasaulyje. Naudojau bei kūriau automatinius testus šiai sistemai ir įrenginiams nuo pat pirmos darbo dienos šioje įmonėje. Tik neseniai pradėjau domėtis, kaip pačiam sukompiliuoti šia programinę įrangą bei padaryti pakeitimus tokius joje.

Taigi, norint kompiliuoti operacinę sistemą, reikia visų pirma reikia pasirinkti teisingą produktą ir susidėti visas tvarkykles, kurių reikia. Kai kurios tvarkyklės būna įrašytos iš pat pradžių. Pirmiausia man reikėjo sužinoti, kaip veikia ir kokių tvarkyklių (angl. drivers) reikia, kad viskas veiktų. Sužinojus ko reikia, galima pradėti jų ieškoti OpenWRT meniu lange. Parašius komandą ‘make menuconfig’ linux terminalo aplinkoje, matomas mėlynas langas su pasirinkimais. Ten yra labai daug įvairių nustatymų. Susiradus visus reikalingus komponentus, sekantis žingsnis yra sukompiliuoti naują programinę įrangą. Pirmą kartą tai darant, užtrunka daugiau nei pusvalandį, sekančius kartus, mažiau nei 10 minučių. Taigi, kai turime naują programinę įrangą mūsų įrenginiui, galime ją išbandyti. Įsirašius programinę įrangą į įrenginį matoma, jog reikalingi komponentai yra surašyti.



6 pav. OpenWRT programinės įrangos tvarkykliu bei nustatymų meniu

Daugiausia darbo su šiuo projektu buvo darant interneto jungčių pakeitimus, kadangi tai jau negali būti pakeista per meniu langą. Šiam, bei kitiems pakeitimams, tokiems kaip LED lempučių konfiguracijai, bei GPIO jungčių automatiniam įjungimui reikia keisti tam tikrą kodą. Vienur reikia pakeisti startavimo programas, kitur reikia pakeisti tiesiog konfiguracijas. Dažniausiai pakeitimai buvo daromi naudojant “shell” programavimo kalbą.

Projektas dabar yra pabaigos stadijoje, kurioje bendraujame su klientu ir atsakinėjame į rūpimus klausimus.

5. Apibendrinimas

5.1 Atlikti darbai

Atliekant anksčiau minėtus darbus buvo įvykdyti visi praktikos tikslai. Darbas komandoje buvo labai svarbus aspektas visų projektų metu. Tiek dirbant su 60 GHz projektu, tiek integruojant LTE modulį į įrenginį reikėjo bendradarbiauti bei kasdien praleisti susirinkimuose po 1-2 valandas. Taip pat pagilinau savo anglų kalbos žinias, kadangi darbas vyko su kanadiečiais, o statuso atnaujinimo susirinkimai vykdavo kasdien.

Darbo metu buvo tiek rašomi automatizuoti testai, kurie padėjo rasti ne vieną kritinę problemą, tiek pildomi klientų reikalavimai programinėje įrangoje. Rašyti programinį kodą teko ir JavaScript, ir Lua, ir Shell programavimo kalbomis. Supratau, jog dokumentacijų skaitymas bei pasiruošimas, planavimas implementacijai yra pats svarbiausias etapas visame projekte. Darbas šioje įmonėje man leidžia gauti vis naujų iššūkių bei suprasti produkto gamybos ciklą.

5.2 Išvados

Visų projektų metu, buvo galimybė išmokti bei pagilinti šias žinias:

- JavaScript programavimo įgūdžiai
- “OpenWRT” sistemos žinių gilinimas
- Kompiuterinių tinklų žinių gilinimas
- TeamCity, Jira bei Perfoce žinių gilinimas
- Wi-Fi 802.11 standarto žinių gilinimas.

Turėjau galimybę dirbti prie kelių skirtingų projektų, kurie visi savaip įdomūs bei susiję su kompiuteriniais tinklais. Mano nuomone, praktikos lūkesčiai buvo pateisinti, nes pavyko parašyti automatinius testus reikiama tema. Be temos supratimo, testų rašymas būtų labai sunkus, todėl dar kartą galiu pasikartoti, kad labai svarbu suprasti, ką reikia padaryti, o tik paskui daryti.