Co-funded by the
Erasmus+ Programme
of the European Union

УНИВЕРЗИТЕТ “Св. КИРИЛ И МЕТОДИЈ” - СКОПЈЕ

ФАКУЛТЕТ ЗА ЕЛЕКТРОТЕХНИКА И ИНФОРМАЦИСКИ ТЕХНОЛОГИИ

- СОФТВЕРСКИ ДЕФИНИРАНА И ВИРТУЕЛНА ИНСТРУМЕНТАЦИЈА -

Автори:

Вон. проф. д-р Живко Коколански

Доц. д-р Томислав Шуминоски

Скопје, 2019

Содржина

Предговор

Книгата Софтверски дефинирана и виртуелниа инструментација претставува универзитетски учебник наменет за истоимениот предмет во рамките на пост-дипломските студии на Факултетот за електротехника и информациски. Предметот се слуша на насоката Метрологија и менаџмент на квалитет во првиот семестар од студиите. За совладување на материјалот од предметот се подразбира дека студентите веќе имаат стекнато значително знаење од областа на електротехниката, виртуелната инструментација и телекомуникациите.

Учебникот има за цел да се разработат концептите за проектирање на софтверски дефинирана и виртуелна инструментација во LabVIEW, како и концептите на мрежна виртуелизација. При совладување на материјата разработена во учебникот на располагање на студентите е и лабораториска опрема која е наменета за продлабочување на практичните знаења при реализацијата на системите.

Книгата е реализирана со подршка на проектот од програмата Erasmus+ на Европската Унија насловен "Иновативен пристап за развој на софтверски-дефинирана инструментација – ITASDI".

1. Софтверски-дефинирана и виртуелна инструментација

Еден од предизвиците со кои денес се соочуваат инженерите е зголемената комплексност на уредите кои тие треба да ги дизајнираат и тестираат. Покрај тоа, золемувањето на сложеноста на уредите е придружено со примена на голем број различни технологии со што се наметнува и потребата од зголемена флексибилност на системите за тестирање. Единствен начин да се задоволат сите овие барања е преку примена на софтверски-дефинирана инструментација која овозможува развој на наменски прилагодени инструменти со добри метролошки карактеристики. Следната генерација на уреди за тестирање мораат да бидат исклучително флексибилни кон можноста за вршење на најразлични тестирања и да овозможат прилагодливи метролошки функции. Во денешно време, софтверски-дефинираната инструментација најчесто се користи во автоматизираните производни процеси, што претставува доказ и потврда на трендовите во развојот на инструментацијата.

Функционалноста на модуларните инструменти се добива преку кориснички-дефиниран софтвер кој вообичаено работи на некој персонален или индустриски компјутер. Од самата дефиниција се врши јасно разграничување на претставата која операторот ја има за мерните инструменти. За разлика од класичните инструменти кои се независни уреди кои се ограничени на карактеристиките декларирани од производителот, софтверски-дефинираната инструментација е програмабилна и нејзините карактеристики зависат од барањата, проектирањето и изведбата на самото системско решение. Улогата која ја има софтверската апликација кон обезбедувањето на флексибилноста кај овие системи е голема. Преку неа инженерите можат брзо и едноставно да се прилагодат на бараните услови за тестирање. Исто така. преку софтверот може да се врши програмирање на одреден модуларен инструмент да се однесува како кориснички-дефиниран и се дава можност за делење на заеднички командни и мерни сигнали. Од друга страна, софтверски-дефинираната инструментација не е доволно обработена и систематски организирана во литературата. Оваа книга има амбиција да даде придонес во овој домен преку разработка на проблематиката инструментација проектирана во LabVIEW и да ги разјасни концептите на мрежна виртуелизација.

1.1 Трендови за развој на софтверски-дефинирана инструментација

Кај класичната инструментација, во најширока смисла, вообичаено производителите дефинираат специфични системски функции за дадена

архитектура и точно дефинирани метролошки функции и карактеристики кои ја ограничуваат примената на инструментите. При користењето на таквата инструментација голем дел од времето се троши на оперирањето со инструментите, долготраен запис на резултатите и нивна документација. Со напредокот на микропроцесорската технологија и нивната примена во мерната техника се овозможи драстично менување на класичните концепти на инструментацијата. Овие трендови овозможија појава на компјутерски-базирани инструменти каде функциските и метролошките карактеристики се дефинираат софтверски. На тој начин се постигнува голема флексибилност при користењето и надградбата на инструментите, што е особено важно во процесот на дизајн и верификација на електрични уреди и апарати од било кој тип.

Вообичаено, процесот на дизајн и верификација започнува со идејно решение кое го интегрира системскиот дизајн и дизајнот на електричните кола. Во оваа фаза, инструментацијата за тестирање сеуште не се применува бидејќи решението сеуште е во концептуална фаза. За дизајн и проверка на електричните кола вообичаено се користат разни софтверски алатки за проектирање и симулација.

Кога ќе се заврши со фазата на дизајнирање и ќе се добие првиот прототипен примерок од електричната опрема, можат да започнат првичните проверки на тестирање. Меѓутоа, доколку не постојат стандардизирани методи и мерни инструменти за тестирање (што е често случај во практиката), тогаш софтверски-дефинираната инструментација комбинирана со други програмабилни инструменти може да биде најдобро и најоптимално решение. Во таков случај всушност тест инженерите имплементираат наменски развиени мерно-аквизициски системи кој најдобро одговараат на барањата за тестирање и понатамошен развој. Во литературата постојат голем број на комерцијални хардверски решенија со кои се овозможува вмрежување на традиционалните инструменти и проширување на нивната функционалност. Меѓутоа, штом еднаш се воспостави архитектурата на така конципираниот систем, дизајнерите мораат да одат еден чекор понатаму со развој на софтверска алатка со која ќе може да се програмираат поврзаните инструменти. Дури и при такви решенија, мора да се води сметка за зависноста од инструментацискатa платформа, комуникациските интерфејси и компатибилноста со други софтверски алатки. Штом еднаш професионално се воспостави таквото решение, мерно-аквизицискиот процес може да се автоматизира преку едноставни софтверски наредби. Дополнително, едноставно се овозможува понатамошната анлиза и меморирање на добиените резултати. На тој

начин, всушност се врши имплементација на софтверски-дефинирана инструментација.

Од техничка гледна точка, идеата за софтверски-дефинирана инструментација не е нова. На пример, пред петнаесетина години три фактори биле критични за преоѓање на модерниот аналоген анализтор на сигнали во дигитален. Најпрво, интеграцијата на повеќе примо-предаватели во еден единствен продукт (на пример паметните телефони) довеле до барање *Radio Frequency* (RF) анализаторите да можат да вршат тестирање на секој расположлив безжичен стандард. Понатамошниот развој на стандардите се одвивал на таков начин што се појавиле барања за RF анализаторите да можат едноставно да се надградуваат. Конечно, развојот на вградливите микрокомпјутерски системи и *Field Programmable Gate Arrays* (FPGA) овозможиле справување со предизвиците околу барањата на RF анализаторите. Една од јасните придобивки на овој пристап е и можноста за извршување на композитни паралелни мерења.

Денес се смета дека софтверски-дефинираната инструментација има голем број предности споредено со традиционалните инструменти. Како прво, имајќи предвид дека мерењата се исчитуваат со помош на специјализирана софтверска апликација, инструментот може едноставно да се надгради да врши други анализи и на друг тип сигнали. Покрај тоа, една од нај јасните придобивки на софтверски-дефинираната инструментација е постигнувањето на голема брзина на мерење. Ако погледнеме во иднината на софтверски-дефинираната инструментација ќе наидеме потреба за развој во две насоки:

1. Поголема моќ за обработка на сигнали, и
2. Поголема мерна апстракција.

Затоа, се очекува дека концептот на софтверски-дефинирана инструментација во следната деценија ќе завземе ново значење за инженерите. Тие ќе можат преку интуитивни графички системи брзо и ефикасно да ги менуваат функциските карактеристики на мерните системи и лесно да ги прилагодуваат на барањата во иднина.

1.2 Виртуелна инструментација

Во поширока смисла, виртуелен инструмент претставува софтверски-контролиран систем за генерална намена чија функционалност се реализира преку персонален компјутер и специјализирани хардверски електронски единици. За постигнување на овие карактеристики, виртуелниот инструмент ги комбинира комерцијалните компјутерски технологии, со лесно приспособлив

софтвер со кој може да се реализираат прототипни системи со карактеристики кои ги дефинира корисникот во зависност од примената. Една од предностите на виртуелните инструменти е можноста за развој на мерно-информациски и мерно-управувачки системи со специфични карактеристики кои вообичаено не се унифицирани кај комерцијалните инструменти, односно неопходно е користење на повеќе наменски комерцијални инструменти за реализација на системот. Друга предност е можноста за искористување на ресурсите на персоналните компјутери, на пример: голем расположлив мемориски простор, можност за примена на голем број компјутерски периферни единици, пристап на интернет и можност за вршење дистрибуирани мерења, едноставно ракување и др. Денес, постојат повеќе софтверски платформи придружени со соодветна хардверска подршка кои овозможуваат развој на виртуелна инструментација. Меѓутоа, најшироко прифатена платформа е LabVIEW (*Laboratory Virtual Instrumentation Engineering Workbench*) од американската компанија *National Instruments*.

Виртуелните инструменти во LabVIEW се состојат од два меѓусебно тесно поврзани модули: преден панел и блок дијаграм. Предниот панел е делот кој овозможува развој на наменски кориснички интерфејс. По правило, сите контроли, индикатори, графици и други единици кои служат за размена на информации помеѓу корисникот и програмата се поставуваат на предниот панел. Предниот панел треба да биде визуелна реална преслика на проблемот кој се решава. За таа намена, виртуелните инструменти нудат можност за дизајн на целосно нови контроли/индикатори или прилагодување на веќе постоечките во облик кој најдобро ја отсликува визуелната перцепција на системот. Генерално, не постои ограничување за начинот на приказ на предниот панел и истиот може едноставно да се прикаже на екранот од персонален компјутер но и на други специјализирани влезно/излезни панели за приказ кои на пример многу често се користат во индустријата. Секој преден панел е придружен со блок-дијаграм. Блок дијаграмот е модулот кој е одговорен за имплементација на програмскиот код и негово организирање за реализација на алгоритамот на работа на програмата. Програмирање на блок-дијаграмот се заснова врз принципите за таканаречено графичко програмирање. Програмирањето се врши преку дефинирање на блоковската структура на мерниот систем и преку дефинирање на нивните карактеристики. На тој начин може многу едноставно да се реализираат сложени матетатички функции и други функции за дигитална обработка на мерниот сигнал. Од друга страна, алгоритамот на работа на програмата се реализира со помош на условени циклуси, структури и сл. Дополнително, програмската платформа

LabVIEW овозможува подршка со други стандардизирани развојни платформи како MATLAB, C++ и сл., со кој може да се реализираат одредени програмски секвенци или да се реискористат истите од некој веќе готов програмски код.

Јасно е дека изнесените карактеристики на виртуелните инструменти имаат голема предност за примена во образованието, и тоа од повеќе причини: студентите се мотивираат да развиваат уникатни решенија на проблемите, имаат можност универзално да ги применат знаењата од други програмски платформи, се пружа можност за проверка на теоретските поставки и нивна реална хардверска експериментална верификација, се дава можност за реализација и тестирање во домашни услови на лични персонални компјутери и др.

2. Системи за аквизиција на податоци во LabVIEW

Во инженерството постојано се јавува потреба од мерење на различни физички величини од електрична и не-електрична природа. Овие физички величини можат да се мерат со помош на независни мерни инструменти или пак да се интегрираат во еден мерен систем со помош на персонален компјутер. Вториот пристап нуди одредени предности при контролата, визуелизацијата и обработката на податоците. Имено, персоналните компјутери се одликуваат со големи мемориски и пресметковни капацитети кои можат да се стават во функција на мерно-информациските и мерно-управувачките системи, а со тоа значително да ги подобрат нивните карактеристики. Ваквите системи кои интегрираат персонални компјутери со цел мерење, прибирање и обработка на податоци се нарекуваат компјутерско-базирани системи за аквизиција на податоци.

Компјутерско-базираните системи за аквизиција во главно содржат два дела: хардверски и софтверски дел. Хардверскиот дел на системот содржи електронски модули кои овозможуваат регистрирање на физичката величина која е предмет на мерење и нејзина преобразба во облик кој е разбирлив за компјутерот. Најпрво физичката величина се регистрира со помош на сензор кој врши преобразба на истата во електричен сигнал (напон, струја или електричен полнеж), или пак во некој негов модулациски параметар (отпорност, капацитивност или индуктивност). Понатаму сигналот се обработува со одредени електронски кола за кондиционирање кои овозможуваат оптимизирање на неговите параметри согласно карактеристиките на влезните канали на картичката за аквизиција на податоци. Конечно, обработениот сигнал се преобразува во дигитален облик со помош на аналогно-дигитален конвертор по што истиот може да се регистрира со персонален компјутер. Покрај сензорот кој секогаш се наоѓа во средината во која се врши мерењето, хардверскиот дел од мерниот систем делумно или целосно се реализира во вид на специјализирани картички за аквизиција на податоци кои се поврзуваат со компјутерот преку стандардизирани комуникациски интерфејси (USB, PCI и др.). Од друга страна, софтверскиот дел од компјутерско-базираните системи за аквизиција на податоци се однесува на корисничкиот интерфејс на мерниот систем. Интеракцијата на корисникот со компјутерот се изведува со помош на наменски развиена софтверска апликација која овозможува управување, манипулирање, обработка, приказ и трајно меморирање на податоците. Фокусот во ова поглавје е поставен на развојот на софтверските апликации на компјутерско-базираните системи за аквизиција на податоци со помош на LabVIEW.

Системите за аквизиција на податоци можат да се категоризираат врз база на различни критериуми. Еден важен критериум за кој ќе стане збор во ова поглавје се однесува на начинот на прибирањето на податоците. Имено, во зависност од барањата и примената на мерниот систем, прибирањето на податоците може да биде синхронизирано со тактот на системот за аквизиција или пак синхронизирано со одредени надворешни временски настани. Аквизицијата синхронизирана со тактот претставува начин за континуирано и униформно прибирање на сигналите независно од какви било други надворешни настани или акции од корисникот. Земањето на одбироците од сигналите се врши периодично и со точно дефинирана фреквенција на семплирање. Од друга страна, синхронизираниот метод на аквизиција со надворешни настани подразбира прибирање на податоците во корелација со други надворешни сигнали или акции од страна на корисникот. Во овој случај аквизицијата на сигналите е апериодична, не-детерминистичка и управувана од таканаречени тригер настани.

2.1 Аквизиција синхронизирана со тактот на системот за за аквизиција на податоци

Времето на извршување претставува основен параметар при дизајнот, управувањето, и тестирањето на хардверски и софтверски мерно-аквизициски решенија. Јасно е дека тоа треба да биде клучен параметар при проектирањето на каков било систем. Преку времето на извршување всушност се дефинираат способностите на решението, но за жал истото е опфатено од голем број влијанија.

Потребно е да се направи разлика помеѓу аквизиција на податоци каде што времето на извршување е дефинирано софтверски или хардверски. Софтверски контролираното време на извршување вообичаено се применува за по бавни апликации, онаму каде што не е потребно постигнување на големи брзини. Од друга страна, хардверски контролираното време на извршување е многу по точно бидејќи се базира на вграден хардверски такт во картичката за аквизиција. За разлика од тоа, при софтверски контролираното време на извршување, управувањето е од страна на тајмерот на оперативниот систем кој во основа е не-детерминистички. Основна препорака е да не се користи софтверска контрола на времето каде што брзината на земање одбироци е поголема од 100 примероци во секунда. Во ова поглавје ќе се обработат техниките за хардверски контролирано време на аквизиција.

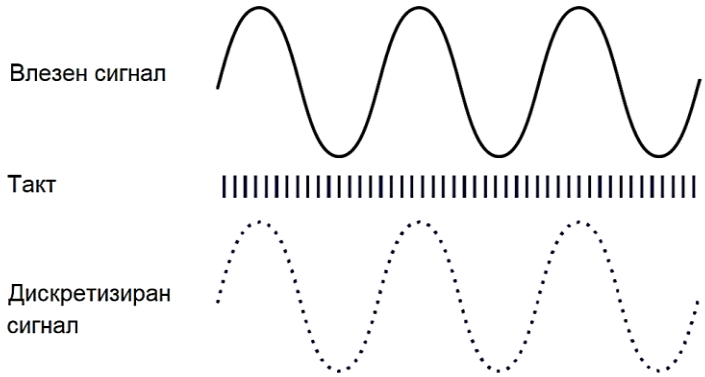
Хардверски контролираното време на аквизиција претставува процес на прибирање на податоци од страна на специјализиран хардвер наречен “картичка за аквизиција”. Кај овој процес, контролата на

брзината на земање одбироци се извршува од страна на внатрешниот такт, најчесто кварцен осцилатор, кој се одликува со голема точност. Покрај тоа, хардверскиот такт вообичаено има многу по голема фреквенција, а со тоа се овозможува постигнување на многу големи брзини на земање примероци и избегнување на преклопување на фреквенцискиот спектар (*aliasing*). Ваквите решенија би биле многу по добри и по точни во споредба со имплементација во која аквизицијата се контролира софтверски. Кај софтверските решенија каде што се користи тајмерот на оперативниот систем можни се голем број на пречки кои ќе доведат до варијации во времето, на пример: работа на други паралелни софтверски апликации, акции од корисникот, зафатеноста на процесорската единица и др. Покрај тоа, постојат хардверски решенија на картичките за аквизиција кои имаат вградени мемориски структури (бафери) во кои може континуирано да се запишуваат серии униформни одбироци од влезните сигнали. Овие одбироци се префрлуваат од мемориските структури во меѓу меморијата на компјутерот пред LabVIEW да изврши нивно исчитување.

Контрола на времето на извршување и техниките за синхронизација овозможуваат координирана работа на разни кориснички-дефинирани временски настани и со тоа значително се зголемуваат можностите кое ги нуди системското решение. При аквизицијата на податоци, овие настани можат да бидат серија одбироци или одбироци од повеќе системи. Во секој случај, временскиот детерминизам е многу важен бидејќи ја помага координацијата или споредбата на прибраните податоци од сигналот во однос на времето. На тој начин сигналите би можеле меѓусебно да се споредуваат.

Напоменавме дека во однос на синхронизацијата при земањето на примероците, техниките за аквизиција можат да се поделат во две категории: аквизиција контролирана од хардверски такт, и аквизиција контролирана од софтверски такт од оперативниот систем. Аквизиција контролирана од хардверски (внатрешен) такт се извршува кога секој одбирок од влезниот сигнал е синхронизиран со тактот на референтен кварцен осцилатор. Тоа значи дека податоците се мерат или генерираат со константна брзина дефинирана од независен такт кој се одликува со добар детерминизам.

Еден пример за аквизиција контролирана од внатрешен такт е прикажан на сликата 2.1 Аквизицијата на сигналот се контролира од страна на референтен такт кој генерира импулси со дадена фреквенција која соодветствува со брзината на земање одбироци од влезниот сигнал.



Сл. 2.1 Аквизиција контролирана од надворешни настани

2.1.1 Примена на виртуелниот инструмент *DAQmx timing*

Со помош на виртуелниот инструмент *DAQmx Timing* може да се врши нагодување на брзината на земање одбироци, бројот на одбироци за мерење или генерирање и по потреба создавање на мемориски бафер. Со виртуелниот инструмент се овозможува дефинирање на соодветниот тип на такт во зависност од примената, на пример: такт за примероци такт за примероци (*Sample Clock*), ракување (*Handshaking*), имплицитно користење (*Implicit*), користење на бранов облик (*Use Waveform*) и детекција на промена (*Change Detection*).

При користење на аналоген влезен канал потребно е да се избере тактот за примероци (*Sample Clock*) од паѓачкото мени на виртуелниот инструмент. Во таков режим, можно е да се дефинираат следните параметри:

- *Sample mode* – дефинира дали виртуелниот инструмент ќе се користи за конечно или континуирано прибирање на податоци.
- *Samples per channel* – се користи за дефинирање на бројот на примероци на влезниот или излезниот канал, доколку режимот за прибирање на податоци е *Finite Samples*. Вредноста на овој параметар ја дефинира големината на баферот кој се користи за пренос на податоците од картичката за аквизиција во LabVIEW.
- *Rate* – е параметар со кој се однесува брзината на одбирање изразена во примероци во секунда. Во случај кога се користи надворешен извор на такт, овој параметар треба да се нагоди на максималната брзина на земање на примероци која се очекува дека надворешниот такт може да ја постигне.

- *Source* – го дефинира изворниот приклучок на тактот (*Sample Clock*). Доколку сакаме да го нагодиме системот да го користи внатрешниот такт на картичката за аквизиција, тогаш потребно е влезот со кој се дефинира овој параметар да остане не поврзан.
- *Active edge* – се користи за дефинирање на активниот раб на тактот (растечки или опаѓачки) во однос на кој се врши мерење или генерирање на сигналот.
- *Task/channels in* – дефинира назив на каналот или пак дефинира листа од виртуелни канали на кои се однесува задачата. Доколку корисникот дефинира листа од канали, тогаш виртуелниот инструмент *NI-DAQmx* автоматски ја дефинира задачата.

Параметарот *Handshaking* на виртуелниот инструмент *DAQmx Timing* го одредува бројот на дигитални примероци кои треба да се добијат или генерираат преку користење на методот на ракување помеѓу картичката за аквизиција и периферниот уред.

Имплицитната инстанца на виртуелниот инструмент *DAQmx Timing* се користи за нагодување на бројот на примероци за мерење или генерирање без при тоа да се дефинира времето на аквизиција. Овој режим вообичаено се користи кога во задачата нема потреба од дефинирање на времето на аквизиција, на пример при користење на бројачи за мерење фреквенција или генерирање на низа од импулси.

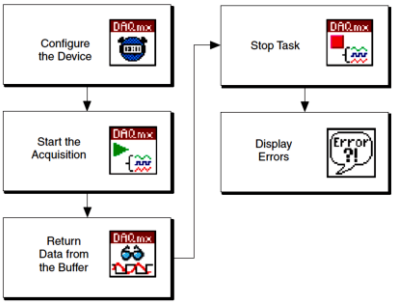
Инстанцата *Use Waveform* на виртуелниот инструмент *DAQmx Timing* се користи за дефиниање на разликата на времето помеѓу одбироците (*dt*) на сигналот за дефинирање на брзината на аквизиција на податоци. Во случај кога се користи режимот *Finite Samples*, виртуелниот инструмент *NI-DAQmx* го пресметува бројот на примероци во сигналот. Важно е да се напомене дека овој виртуелен инструмент не се користи за генерирање на податоци. За да се постигне тоа, виртуелниот инструмент треба да се поврзе со *DAQmx Write*.

2.1.2 Конечна аквизиција со мемориски бафер

Конечната аквизиција со мемориски бафер се користи за едновремено прибирање на повеќе одбироци од влезниот сигнал. Основниот концепт за реализација на овој метод на аквизиција е илустриран на сл. 2.2. Она што е потребно при конфигурирање на аквизицијата е да се нагодат бројот на примероци и брзината на земање на примероците.

Виртуелниот инструмент *DAQmx Timing* се користи за конфигурирање на времето на аквизицијата и меморискиот бафер на

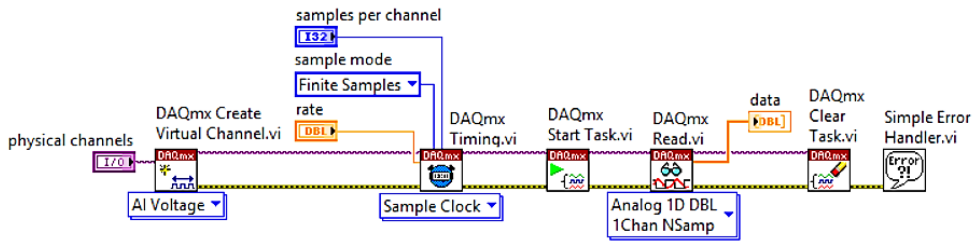
картичката за аквизиција која што се користи. Потоа, виртуелниот инструмент за исчитвување (*DAQmx Read*) чека се додека сите примероци од сите канали бидат на располагање за исчитување. Виртуелниот инструмент понатаму ги предава податоците на следната функција од програмата. Виртуелниот инструмент *DAQmx Stop Task* се користи за запирање на задачата и за ослободување на алоцираните ресурси кои се доделени на картичката за аквизиција. Конечно, виртуелниот инструмент за справување со грешки (*Error Handler*) врши приказ на било која грешка која евентуално настанала во процесот на аквизиција.



Сл. 2.2 Тек на програмата при конечна аквизиција со мемориски бафер

Блок дијаграмот прикажан на сл. 2.3 го илустрира начинот за извршување на конечна аквизиција на податоци. Процесот започнува со нагодување на режимот на аквизиција и брзината на земање на примероци. За прибирање на конечен број на примероци, режимот за аквизиција на податоци треба да биде нагоден на конечна аквизиција (*Finite Samples*). Потоа се нагодува влезниот податок за брзината на земање одбироци (*rate*) и бројот на примероци по канал (*samples per channel*).

Во следниот чекор виртуелниот инструмент започнува аквизицијата на податоците. Од тој момент програмата го чека виртуелниот инструмент за исчитување се додека не се наполни мемориskиот бафер. Тогаш кога баферот ќе биде полн, виртуелниот инструмент ги исчитува податоците од него и програмата продолжува со извршување. Согласно примерот прикажан на сл. 2.3, *DAQmx Clear Task* ја запира аквизицијата, а виртуелниот инструмент за справување со грешки (*Simple Error Handler*) ги прикажува евентуалните грешки кои настанале при аквизицијата.



Сл. 2.3 Изведба на виртуелен инструмент за конечна аквизиција со мемориски бафер

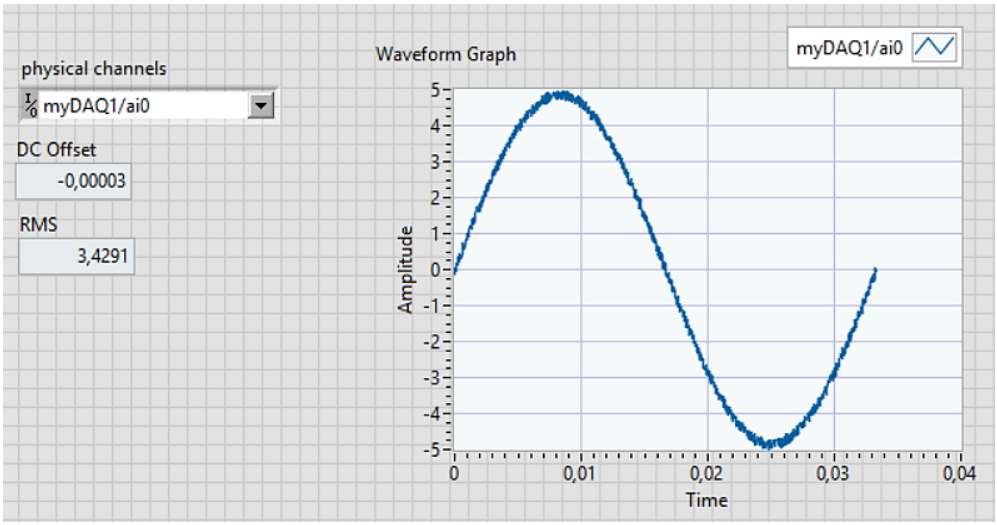
Во примерот прикажан на сликата, влезниот податок за дефинирање на број на примероци по канал не е поврзан. Од овие причини, автоматски се пресметува баојот на примероци потребни за исчитување врз база на нагодувањата на виртуелниот инструмент *DAQmx Timing*. Всушност, бројот на примероци по канал се нагодува на вредност –1. Потоа виртуелниот инструмент за исчитување враќа дво-димензионална низа од податоци кои можат директно да се поврзат на график за сигнали. Меѓутоа, низата не содржи информација за времето на податоците, каков што е случајот со податочниот тип за сигнали (*waveform*).

Во согласност со вообичаените програмски практики во LabVIEW, кластерот за грешка потребно е да се поврзе за сите виртуелни инструменти со цел контрола на потокот на податоците. Во случај кога ќе се појави грешка во виртуелниот инструмент *DAQmx Start Task*, *DAQmx Read* или *DAQmx Stop Task*, соодветниот виртуелен инструмент враќа информација за грешката низ кластер терминалот и не се извршува. На пример, да претпоставиме дека се појавила грешка во виртуелниот инструмент *DAQmx Start Timing*. Тогаш *DAQmx Start Timing* запира со извршување и ја предава информацијата за грешка на виртуелниот инструмент *DAQmx Start Task*. Исто така *DAQmx Start* не се извршува и ја предава информацијата за грешка понатаму се додека не стигне до виртуелниот инструмент за справување со грешки (*Error Handler*) кој ја прикажува грешката.

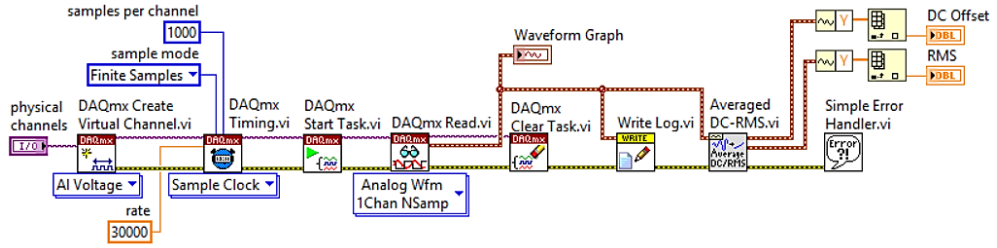
При аквизиција на аналоген сигнал, пред процесирањето тој може да се засили со вградениот инструментациски засилувач и тој сигнал да се проследи до аналогно-дигиталниот конвертор. Потоа, сигналот вообичаено се проследува низ FIFO мемориска структура која ги запишува податоците до моментот кога истите се пренесуваат кон персоналниот компјутер преку *Direct Memory Access* (DMA) или *Interrupt Request* (IRQ).

Виртуелниот инструмент кој е опишан во следниот пример демонстрира аквизиција на податоци од реален физички влезен сигнал.

Со виртуелниот инструмент се врши одредување на ефективната вредност и еднонасочното поместување на сигналот. Резултатите од мерењето потоа се запишуваат во датотека за логираање. Предниот панел на виртуелниот иснструмент е прикажан на сл. 2.4, додека соодветниот блок дијаграм е прикажан на сл. 2.5.

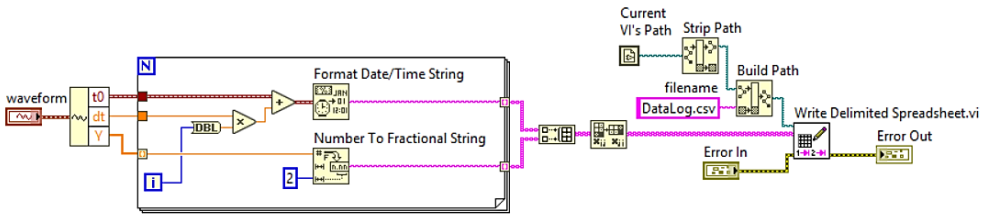


Сл. 2.4. Преден панел на конечна аквизиција со мемориски бафер



Сл. 2.5. Блок дијаграм на реализација на виртуелен инструмент за конечна аквизиција со мемориски бафер

Во примерот прикажан на сл. 2.4 и сл. 2.5 се прикажува конечна аквизиција на сигнал од кој се добиваат рамки од 1000 примероци со брзина од 30000 примероци во секунда од дефинираниот влезен канал. Добиените податоци кои се добиваат од виртуелниот инструмент за исчитување во сигнален податочен тип се прикажуваат на соодветен график. Потоа, се врши бришење на задачата и податоците се запишуваат во датотеката *Write Log.vi*. Реализацијата на под-виртуелниот инструмент за запис во датотека е прикажан на сл. 2.6. Конечно, со примена на виртуелниот инструмент *Averaged DC-RMS* програмата врши пресметка на ефективната вредност и еднонасочното поместување.

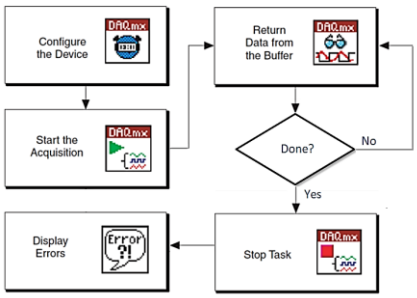


Сл. 2/6. Пример за конечна аквизиција со мемориски бафер и запишување во датотека за логирање

Датотеката за логирање е наречена *DataLog.csv* и истата содржи две колони податоци. Првата колона содржи информација за времето на исчитување додека втората колона се однесува на амплитудата на аналогниот сигнал кој се мери. За запис во датотеката се користи табеларно распределен запис кој едноставно може да се прочита од друг тип програми.

2.1.3 Континуирана аквизиција со мемориски бафер

Основната разлика помеѓу конечната и континуираната аквизиција со мемориски бафер се однесува на бројот на примероци кои се прибираат. Кај конечната аквизиција, бројот на примероци е точно дефиниран, додека кај континуираната аквизиција примероците се прибираат постојано и неограничено. Процесот на континуираната аквизиција може да се претстави со дијаграмот прикажан на слика 2.7.



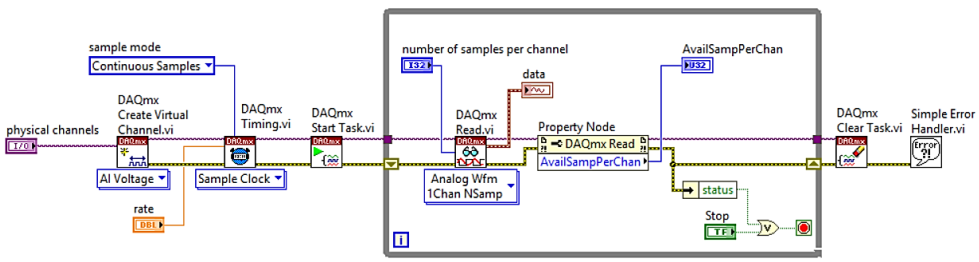
Сл. 2.7. Постапка на континуирана аквизиција со мемориски бафер

Првите три чекори од дијаграмот се идентични како и чекорите на конечната аквизиција со мемориски бафер. Се започнува со нагодување на картичката за аквизиција, започнување на аквизицијата, и подготовка за исчитување на податоците. Меѓутоа, имајќи предвид дека во овој случај податоците непрекинато се прибираат, јасно е дека истите треба и непрекинато да се исчитуваат со некој итеративен циклус. Циклусот може да заврши со извршување при евентуална појава на грешка во процесот на аквизиција, или пак со дадена акција на корисникот преку

предниот панел на виртуелниот инструмент. Во случај кога нема појава на грешки и команда за запирање, секоја итерација од циклусот врши исчитување на податоците. По завршување на циклусот, задачата за аквизиција се запира и се ослободуваат сите зафатени ресурси. По правило, на крајот можат да се прикажат евентуалните грешки со примена на *Simple Error Handler*.

Блок дијаграмот на виртуелниот инструмент за континуираната аквизиција со мемориски бафер е прикажана на сл. 2.8. Може да се забележи дека нема големи разлики споредено со конечната аквизиција. Разликите се состојат во следното::

- Виртуелниот инструмент за исчитување се наоѓа во рамките на while циклус.
- Влезниот терминал за број на примероци по канал може да се дефинира од корисникот. Кај конечната аквизиција, виртуелниот инструмент *NI-DAQmx* автоматски го одредува бројот на примероци. Доколку влезниот терминал за број на примероци по канал се остави не поврзан (нагоден на -1), тогаш *NI-DAQmx* врши исчитување на сите примероци кои се на располагање во меморискиот бафер.
- Бројот на расположливи примероци по канал (*backlog*) може да се следи.



Сл. 2.8. Приказ на блок дијаграм за континуирана аквизиција со мемориски бафер

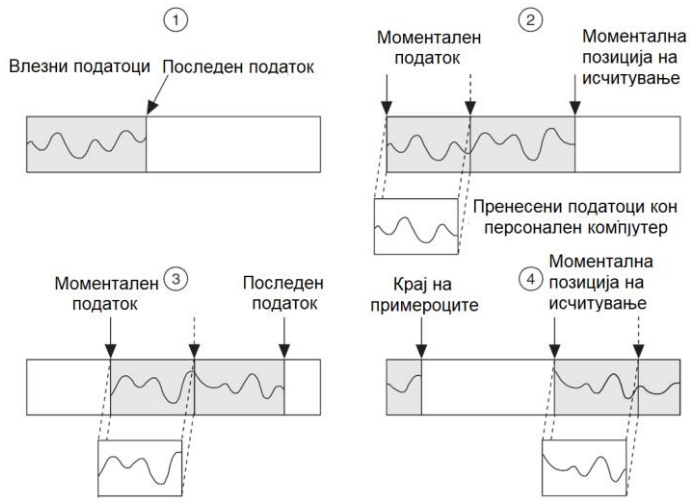
Континуираната аквизиција започнува со нагодување на тактот, режимот на аквизиција, број на примероци по канал (должината на меморискиот бафер) и брзината на земање на примероците. Во случај ако терминалот за број на примероци по канал не се поврзе, тогаш автоматски се нагодува должината на меморискиот бафер врз база на брзинта на земање на примероците. По првичните нагодувања, виртуелниот инструмент *DAQmx Start* ја активира аквизицијата на податоците.

Виртуелниот инструмент за исчитување е сместен во рамките на итеративен циклус. Секоја итерација од циклусот се врши исчитување на примероците складирани во меморискиот бафер. За да се спречи прелевање на баферот, бројот на примероци по канал кои се исчитуваат не може повеќе да е е днаков со должината на меморискиот бафер. Добра практика е да се нагоди бројот на примероците по канал да биде една четвртина до една половина од должината на баферот. Исто така, бидејќи LabVIEW постојано испраќа податоци во баферот, важно е да се врши преглед на расположливите примероци во баферот со цел да се провери дали истиот се празни доволно брзо.

Доколку бројот на расположливи примероци по канал има тенденција на зголемување, тогаш баферот може да прелее и да генерира информација за грешка. Со информацијата за грешка, или со акција од страна на корисникот се запира виртуелниот инструмент за исчитување кој е поставен во рамките на циклусот. Со запирање на циклусот, виртуелниот инструмент *DAQmx Stop* ја запира задачата и ги ослободува зафатените ресурси, додека Simple Error Handler врши приказ на појавените грешки.

2.1.4 Кружен мемориски бафер

Функционирањето на меморискиот бафер во континуиран режим за аквизиција не е едноставен процес бидејќи компјутерот користи еден единствен бафер кој не може да задржи повеќе податоци. За да се приберат повеќе податоци од должината на баферот мора да се користи таканаречен кружен бафер. На сл. 2. 9 е илустриран принципот на работа на кружниот бафер.



Сл. 2.9. Принцип на работа на кружен мемориски бафер

Кружниот мемориски бафер е многу сличен на вообичаениот бафер. Разликата е во тоа што кога ќе се дојде до крајот на кружниот бафер наместо да се запре се започнува од почеток. Се започнува со дефинирањето на должината на баферот при нагодување на бројот на примероци по канал со виртуелниот инструмент. Во моментот кога виртуелниот инструмент *DAQmx Start Task* ја започнува аквизицијата баферот започнува да се полни. Се разбира дека аквизицијата се случува во рамките на итеративен софтверски циклус.

Нека претпоставиме дека бројот на примероци по канал е нагоден помеѓу една четвртина до една половина од должината на баферот. Во моментот кога бројот на примероци по канал во баферот ќе биде еднаков со бројот на примероци од виртуелниот инструмент за исчитување (*DAQmx Read*) се врши пренесување на примероците во LabVIEW програмата. Истовремено, функцијата *DAQmx Read* поставува знамеце на моменталната позиција во баферот за да може понатаму да врши исчитување од тоа место.

Во меѓувреме, баферот продолжува да се полни со податоци. Виртуелниот инструмент за исчитување исто така продолжва да ги исчитуива податоците од баферот и да ги префрлува во LabVIEW. Во моментот кога баферот ќе се наполни со податоци, новите податоци започнуваат да се запишуваат на почетокот од баферот. Разликата помеѓу знаменцето за крајот на податоците и моменталната позиција е еднаков на бројот на расположливи примероци по канал. Она што би требало да се обезбеди е LabVIEW да го осчитува податоците доволно брзо што ќе спречи знаменцето за крајот на податоците да ја престигне моменталната позиција, во спротивно ќе настане пребришување на старите податоци со нови податоци и LabVIEW ќе испрати сигнал за грешка.

Една од нај честите грешки која може да се појави при користењето на кружниот бафер е грешката за пребришување на податоците. Тоа се случува доколку LabVIEW не е во состојба да ги исчитува податоците од баферот доволно брзо. Меѓутоа, постојат неколку начини кои можат да помогнат до спречување на оваа грешка, но нивната ефикасност и применливост зависи од специфичноста на системското решение.

- Преку зголемување на бројот на примероци по канал (димензијата на баферот). Но, ова решение не е ефикасно доколку празнењето на баферот не е доволно брзо. Треба секогаш да ја применуваме добрата програмска практика за бројот на примероците по канал да биде околу една четвртина до една половина од должината на баферот.

- Зголемување на брзината на празнење на баферот преку зголемување на бројот на примероци за исчитување во виртуелниот инструмент за исчитување. Овој параметар не треба да биде премногу голем бидејќи програмата ќе ја чека функцијата *DAQmx Read* се додека бројот на примероци по канал не биде еднаков на бројот на примероци во виртуелниот инструмент за исчитување. Ова време кое се троши на чекање може да се искористи за празнење на баферот.
- Преку намалување на брзината на земање примероци во виртуелниот инструмент *DAQmx Timing*. На овој начин се врши забавување на брзината со која се врши полнење на баферот со податоци, но ова решение можеби не е применливо доколку системското решение наметнува да се постигнат одредени брзини на мерење.
- Да се избегнување забавување на циклусот во кој се врши исчитувањето со други непотребни анализи и операции.

Друг проблем кој може да се појави при континуираната аквизиција со мемориски бафер се однесува на преполнување на FIFO баферот во картичката за аквизиција. Преполнувањето на овој бафер не е толку често колку преполнувањето на баферот во компјутерот и не постои едноставно решение за негово надминување. Ваквиот проблем се појавува во случај кога FIFO баферот не се празни доволно брзо. Овој бафер зависи од DMA или IRQ механизмите за да се изврши пренос кон баферот во компјутерот. Постојат само неколку можни пристапи за да се спречи појава на оваа грешка:

- Примена на DMA за пренос на податоци (доколку постои) бидејќи тој е значително по брз во споредба со IRQ механизмот.
- Преку намалување на брзината на земање одбироци по канал во виртуелниот инструмент *DAQmx Timing*.

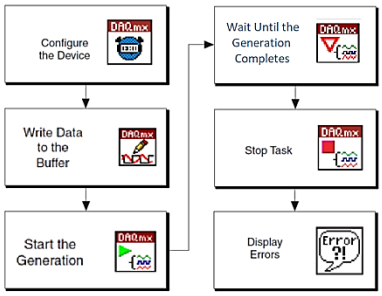
2.1.5 Генерирање на сигнал со конечен број примероци и мемориски бафер

Генерирањето на сигнал со конечен број примероци и мемориски бафер е режим на работа во кој програмата генерира низа од аналогни излезни примероци со дадена брзина кои физички се појавуваат во вид на континуални променливи сигнали. Слично како и кај аквизицијата, овој режим на работа исто така може да се постигне во конечен број точки или со континуирано генерирање. Во секој случај, двата пристапи се одликуваат со два основни чекори:

- Запишување на примероците во бафер. Примероците се превземаат од LabVIEW програмата во меѓу-меморијата и потоа се испраќаат до картичката за аквизиција која подржива генерирање на излезни аналогни сигнали.
- Пренос на примероците од баферот до картичката за аквизиццција. Брзината со која се пренесуваат примероците зависи од специфични временско-зависни параметри.

Кај хардверски-контролираното генерирање, сигналот се управува од стабилен такт добиен од кварцен осцилатор вграден во картичката за аквизиција. Овој такт може да работи со многу поголема фреквенција (брзина) отколку софтверски-генерираниот такт. Тоа значи дека на таков начин можат да се изврши генерирање на сигнали со многу поголема горна гранична фреквенција.

На сл. 2.10 е прикажан дијаграм кој опишува генерирање на сигнал со конечен број примероци и мемориски бафер.

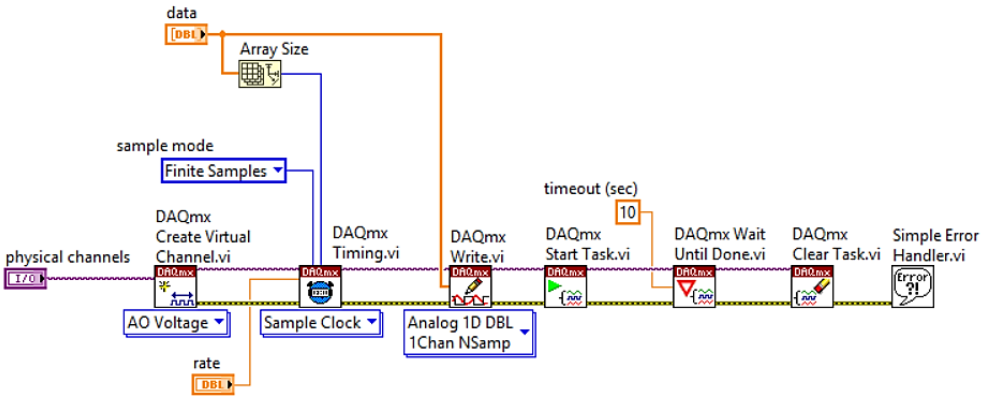


Сл. 2.10 Принцип на работа на режим на генерирање на сигнал со конечен број примероци и мемориски бафер

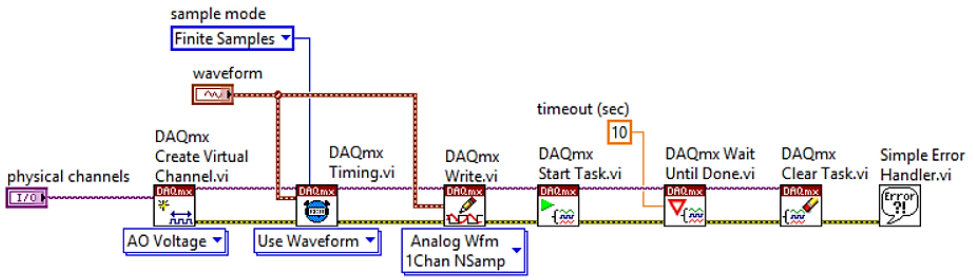
За изборот на режимот на работа во поглед на контрола на тактот (хардверска или софтверска) се користи виртуилниот инструмент *DAQmx Timing*. Со избор на инстанцата *Sample Clock* на виртуелниот инструмент Timing, се врши нагодување тактот при генерирањето на сигналите да се добива од картичката за аквизиција. Дополнително, виртуелниот иснтрумент *DAQmx Timing* ја содржи инстанцата за генерирање бранов облик (*Use Waveform*). Оваа инстанца ја користи временската разлика помеѓу два примероци (*dt*) за пресметка на брзината на генерирање на примероците. На таков начин се воспоставува детерминистичко хардверски-дефинирано одредување на временските настани во кои ќе се генерираат примероците на излезниот сигнал. Важно е да се напомене дека инстанцата *Use Waveform* не врши генерирање на излезниот сигнал, туку само се користи за нагодување на

тактот за излезниот сигнал. За да се генерира излезен сигнал потребно е да се искористи виртуелниот инструмент *DAQmx Write*.

На сликите 2.11 и 2.12 се прикажани блок дијаграми на решение за генерирање со конечен број примероци и мемориски бафер со два различни начини на дефинирање на времето. На блок дијаграмот на сл. 2.11 може да се примети дека се користи низа од децимални броеви за претстава на примероците од излезниот сигнал и нагодување на времето со виртуелниот инструмент Timing, додека на сл. 2.12 е прикажано решене во кое временските моменти се превземаат од сигналниот податочен тип.



Сл. 2.11: Блок дијаграм на виртуелен инструмент за генерирање со конечен број примероци и мемориски бафер



Сл. 2.12. Блок дијаграм на виртуелен инструмент за генерирање со конечен број примероци и мемориски бафер каде временските моменти се дефинирани преку сигнален податочен тип

Постојат неколку разлики помеѓу двата претставени начини на генерирање на излезни сигнали - преку низа од децимални броеви и со сигнален податочен тип. Виртуелниот инструмент *DAQmx Timing* има две инстанци кои можат да се искористат за генерирање излезен аналоген сигнал - *Sample Clock* и *Use Waveform*. Имајќи предвид дека се

разгледува генерирање со конечен број примероци, режимот на работа на двете инстаци се нагодува на *Finite Samples*. При користење на инстанцата *Sample Clock*, исто така се нагодуваат и брзината на генерирање и бројот на примероци на излезниот сигнал. Преку дефинирање на бројот на примероци истовремено се дефинира и должината на излезниот бафер. Од друга страна, инстанцата *Use Waveform* се користи еноставно со поврзување на сигнален податочен тип на влезниот терминал. Во тој случај оваа инстанца врши пресметка на брзината на генерирање и бројот на излезни примероци преку параметрите на сигналниот податочен тип.

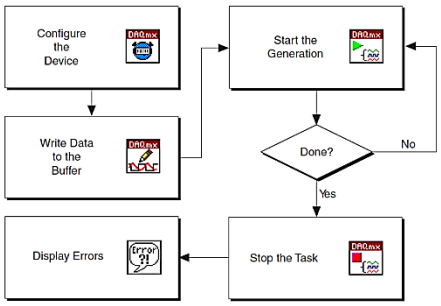
За физичко генерирање на сигналот се користи виртуелниот инструмент *DAQmx Write* кој софтверски-генерираните податоци ги испраќа до баферот на картичката за аквизиција. Оваа функција може да прими два вида на дефиниции за сигналот: низа од децимални податоци или пак сигнален податочен тип. За користење на инстанцат а *Use Waveform* потребно е да се избере сигнален податочен тип од паѓачкото мени на функцијата за генерирање на сигналот (*Write*). Исто така, истиот сигнален податочен тип кој што се користи за дефинирање на временските карактеристики може истовремено да се поврзе и на функцијата за запишување (генерирање). Од друга страна, доколку се користи вградениот такт за дефинирање на временските карактеристики на сигналот потребно е да се избере низа од децимални броеви од паѓачкото мени на виртуелниот инструмент за запишување. Во тој случај, низата од децимални броеви треба истовремено да се поврзе на податочнио влезен терминал data на виртуелниот иснтрумент за запишување (*Write*).

Виртуелниот инструмент *DAQmx Start* се користи за започнување на процесот на генерирање. Функцијата *DAQmx Wait Until Done* чека се додека не се заврши задачата за генерирање или додека не се појави информација за изминато време (*timeout*). Доколку се појават кои било од овие два настани, се активира функцијата *DAQmx Stop Task* која ја запира задачата и процесот на генерирање. За контрола на потокот на податоците и за пренесувае на информацијата за грешка сите виртуелни инструменти се поврзуваат со кластерот за грешка.

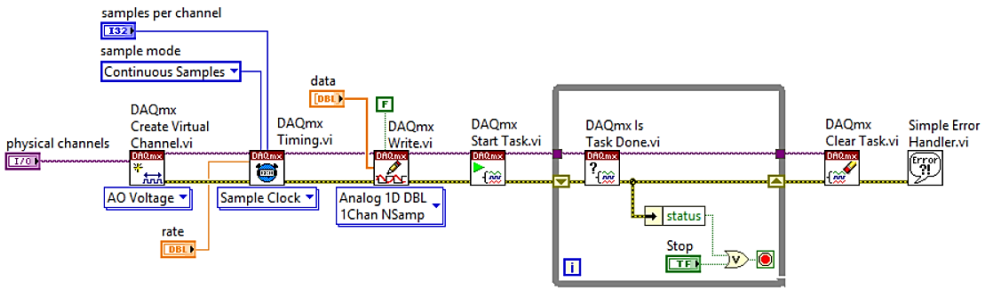
2.1.6 Континуирано генерирање со мемориски бафер

Основната разлика помеѓу конечното генерирање со мемориски бафер во споредба со континуираното генерирање е во бројот на примероци кои се генерираат. Кај конечното генерирање, податоците се генерираат со примена на баферот во конечни секвенци. За разлика од тоа, кај континуираното генерирање податоците се генерираат постојано.

На сл. 2.13 е прикажан процесот на континуираното генерирање со примена на мемориски бафер.



Сл. 2.13. Континуирано генерирање со мемориски бафер



Сл. 2.14 Блок дијаграм на виртуелен инструмент за реализација на континуирано генерирање со мемориски бафер

Блок дијаграмот прикажан на сл. 2.14 е сличен со примерот со конечното генерирање, постојат само неколку разлики: влезниот канал sample mode на виртуелниот инструмент *DAQmx Timing* се нагодува на *Continuous Samples* и виртуелниот инструмент *DAQmx Is Task Done* се поставува во итеративен циклус наместо примена на функцијата *DAQmx Wait Until Done*.

Процесот започнува со нагодувањње на виртуелниот канал и на временските карактеристики преку примена на виртуелните инструменти *Create Virtual Channel* и *Timing*. Потоа, податоците се запишуваат во баферот со примена на виртуелниот инструмент за запишување и се активира извршувањето на задачата. Во меѓувреме итеративниот циклус се користи за проверка на статусот на задачата, дали таа е завршена или не. Процесот на генерирање може да се запре доколку дојде до појава на грешка, или пак ако програмата се запре преку корисничка акција на предниот панел. Со завршување на циклусот, се врши запирање на задачата и приказ на информациите за евентуални грешки кои настанале.

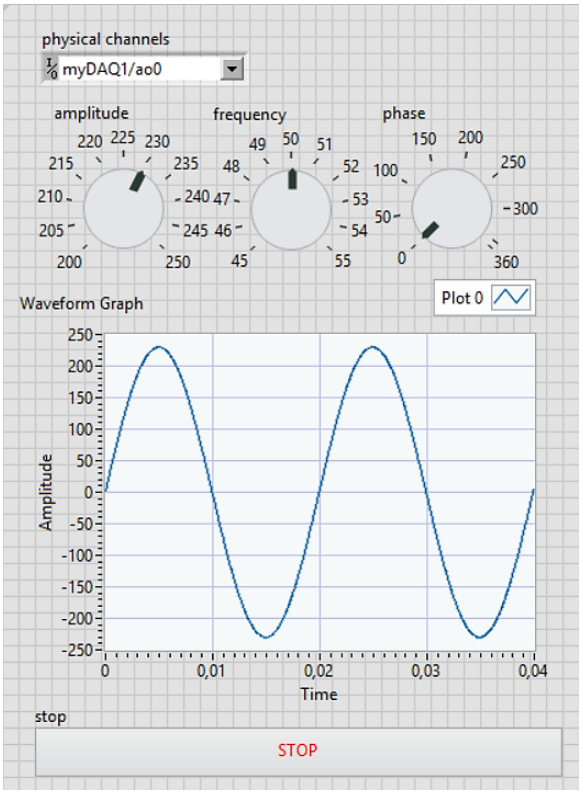
Процесот на последователно генерирање на истите податоци се нарекува регенерирање. Програмата во LabVIEW може да биде нагодена да овозможи или да забрани регенерирање со соодветно конфигурирање на атрубутот *Regeneration Mode*. Тогаш кога овој атибут е оневозможен, за да се изврши континуирано генерирање неопходно е постојано да се запишуваат нови податоци со картичката за аквизиција.

Регенерирањето на податоците исто така може да се изврши и со нагодување на атибутот *Use Only Onboard Memory*. Доколку се овозможи овој атрибут, тогаш податоците само еднаш се пренесуваат до картичата за аквизиција и потоа тие независно се регенерираат. Меѓутоа, доколку во таков режим се обидеме да запишеме нови податоци во картичката за аквизиција тогаш ќе дојде до појава на грешка. Исто така, важно е количината на податоците кои ги запишуваме во картичката за аквизиција со цел нивно регенерирање да соодветствуваат со нејзините мемориските капацитети.

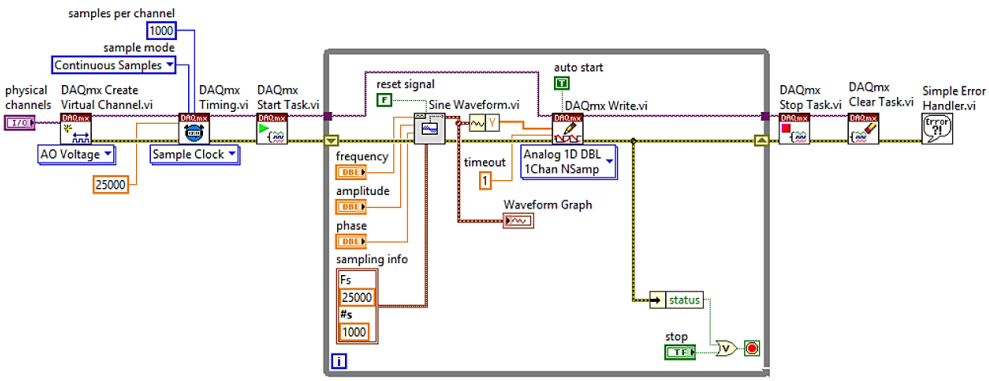
Во случај кога атрибутот *Use Only Onboard Memory* е оневозможен, тогаш потребно е да се врши континуиран пренос на податоци во мемориският бафер дури и во случај кога податоците не се менуваат. Овој начин на регенерирање понекогаш е наречен регенерирање преку меморијата на персоналниот компјутер. Тогаш кога се користи овој начин на регенерирање исто така треба да се нагоди атрибутот *Data Transfer Request Condition* со цел дефинирање на начинот на пренос на податоците од баферот до картичката за аквизиција.

Новите податоци можат да се запишат во баферот со секоја итерација на циклусот. Виртуелниот инструмент за запис треба да се постави во рамките на циклусот и податоците кои треба да се генерираат треба да се поврзат на влезниот терминал data. Во таков случај, потребно е да се дадат на располагање нови податоци за запишување доволно брзо за да се спречи баферот да регенерира стари податоци. Може да се каже дека овој случај е сличен со вршењето на континуирана аквизиција со мемориски бафер. Податоците од баферот мора да се читаат доволно брзо за да се спречи презапишување на баферот. Ваков случај на континуирано запишување со мемориски бафер ќе биде разгледан во следниот пример.

Виртуелниот инструмент кој ќе биде опишан во следниот пример се користи за континуирано генерирање на сигнал со синусен бранов облик, но при тоа корисникот има можност за нагодување на неговата амплитуда, фреквенција, почетна фаза и еднонасочно поместување. Предниот панел на виртуелниот инструмент е прикажан на сл. 2.15 додека блок дијаграмот на решението е прикажан на сл. 2.16.



Сл. 2.15 Преден панел на виртуелен инструмент со континуирано генерирање и мемориски бафер



Сл. 2.16 Блок дијаграм на виртуелен инструмент со континуирано генерирање и мемориски бафер

Виртуелниот инструмент треба да овозможи репродуцирање на физички сигнал на соодветниот расположлив канал од картичката за аквизиција која е на располагање. Виртуелниот инструмент е нагоден да врши генерирање на податоците со брзина од 25000 примероци во секунда. Со секое повикување на виртуелниот инструмент за

запишување се репродуцираат 1000 примероци од сигналот. Сигналот со синусен бранов облик се добива со користење на виртуелниот инструмент *Sine Waveform* кој е поставен во итеративен циклус. Со помош на овој виртуелен инструмент корисникот може да врши нагодување на амплитудата, фреквенцијата и фазта на сигналот. Откако ќе се извлечат амплитудните вредности на сигналот се добива еднодимензионална низа од децимални броеви кои се репрезент на сигналот кој ќе се генерира. Овој сигнал истовремено се прикажува на соодветен графички индикатор. Временските карактеристики на сигналот се дефинираат преку кластерот sampling info input на виртуелниот инструмент. Важно е да се напомене дека програмата може да работи коректно само доколку константата F_s од кластерот е еднаква со брзината на генерирање поврзана на влезот на виртуелниот инструмент *DAQmx Timing*. Вака реализираната програма ќе работи постојано се додека корисникот не притисне на копчето *Stop* или доколку во текот на извршување на програмта дојде до појава на некаква грешка. По завршување на програмата, задачата се запира, се ослободуваат зафатените ресурси и се прикажуваат можните грешки кои настанале.

2.2 Аквизиција синхронизација со надворешни временски настани и сигнали

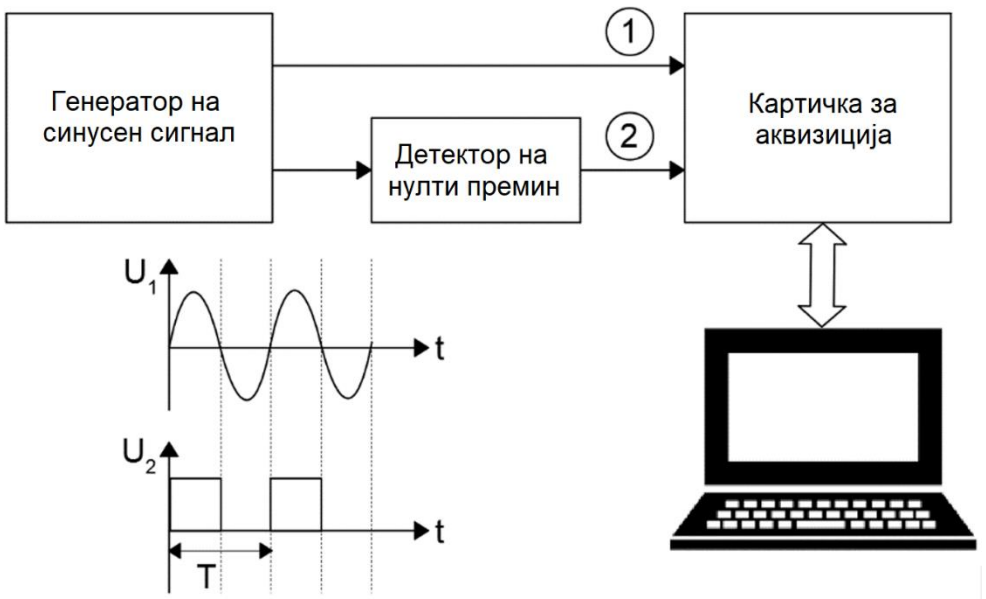
Синхронизираната аквизиција на податоци со надворешни сигнали е контролирана од тригер настани. Тригер претставува сигнал или акција која го дефинира моментот на започнување на аквизицијата. При дефинирањето на тригерот од суштинска важност е да се одговори на две прашања: каква акција треба да предизвика тригерот, и како да се генерира истиот. Меѓутоа, покрај дефинирањето на акцијата потребно е да се одреди и изворот на тригерот. Овој извор може да биде дигитален сигнал, аналоген сигнал или пак акција на корисникот. Во зависност од изорот на тригерот се користат соодветни канали на картичката за аквизиција на податоци (аналогни или дигитални) и се применуваат соодветни програмски архитектури.

2.2.1 Тригер од дигитален раб

Дигиталниот сигнал има две дискретни нивоа (ниско и високо ниво) чии физички напонски граници се дефинирани според станрадизирани нивоа (TTL, CMOS и сл.). Овие сигнали вообичаено се поврзуваат со картичката за аквизиција преку дигитални влезни порти. При промената на дигиталниот сигнал помеѓу двете логички состојби истиот генерира растечки односно опаѓачки раб. Ваквите рабови можат да се искористат за генерирање на тригер настани. Постојат неколку акции кои тригерот може да ги изврши:

- *Reference trigger* – дефинира референтна точка од која започнува аквизицијата на податоци;
- *Start trigger* – започнува процес на аквизиција или генерирање;
- *Pause trigger* – врши паузирање на аквизицијата;

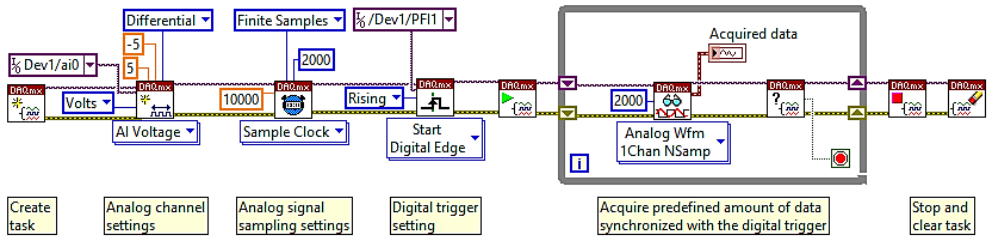
За да го разјасниме процесот на аквизиција со примена на тригер од дигитален раб ќе ја разгледаме проектната задача прикажана на следната слика:



Сл. 2.17 Концепт на систем за примена на тригерирање со дигитален раб. Системот аквизира простопериодичен синусен сигнал со помош на дигитален тригер сигнал кој се генерира со детектор на нулта вредност.

Разгледуваме простопериодичен аналоген сигнал со периода T кој треба да се обработи со компјутер. Од суштинска важност е почетокот на аквизицијата да содветствува со преминот на нултата вредност на сигналот. За да се постигне тоа, во системот прикажан на сликата 2.17 се користи детектор на нулта вредност. Овој детектор генерира дигитален напонски импулс кој соодветствува со ширината на позитивната полуперида на сигналот. На тој начин се добива дигитален сигнал кој има иста периода и фаза како аналогниот сигнал, при што истиот може да се користи како тригер сигнал. Доколку се користи растечкиот раб на дигиталниот сигнал тогаш аквизицијата на сигналот ќе започне со позитивната полуперида на сигналот, а доколку се користи опаѓачкиот раб тогаш аквизицијата ќе започне со негативната полуперида. Примерот прикажан на сликата 2.17 може да најде голема примена во

праксата. На пример, таков пристап е вообичаен при анализа на квалитетот на електричната енергија кај електроенергетските мрежи. Параметарите кои го дефинираат квалитетот на електричната енергија се регулирани со стандардот EN50160, додека нивната пресметка со стандардот IEC 61000-4-30. Споменатите стандарди препорачуваат аквизиција на десет периоди од сигналот на мрежата за пресметка на параметрите од интерес. Дополнително, голем број на алгоритми за класификација на изобличувањата бараат прозорецот од десет периоди на сигналот да започнува со нулта вредност и со позитивен поларитет. Сигналот од мрежата на почеток се обработува со специјализирани електронски кола за аквизиција кои вршат слабеење, галванска изолација и филтрирање на сигналот. Конечно се добива аналоген сигнал со референтни напонски нивоа во границите од ±5 V со номинална фреквенција од 50/60 Hz и ограничен фреквенциски спектар до 3 kHz. Основната реализација на виртуелен инструмент за синхронизирана аквизиција со тригер од дигитален сигнал кој ги исполнува дефинираните барања е прикажана на слика 2.18.



Сл. 2.18 Реализација на виртуелен инструмент во LabVIEW со дигитален тригер на растечки раб

Постапката за аквизиција започнува и завршува со креирање и бришење на задача. Задачата мора да има уникатно и единствено име при секое нејзино повикување, во спротивно програмата ќе врати грешка. Потоа се врши нагодување на карактеристиките на аналогниот канал:

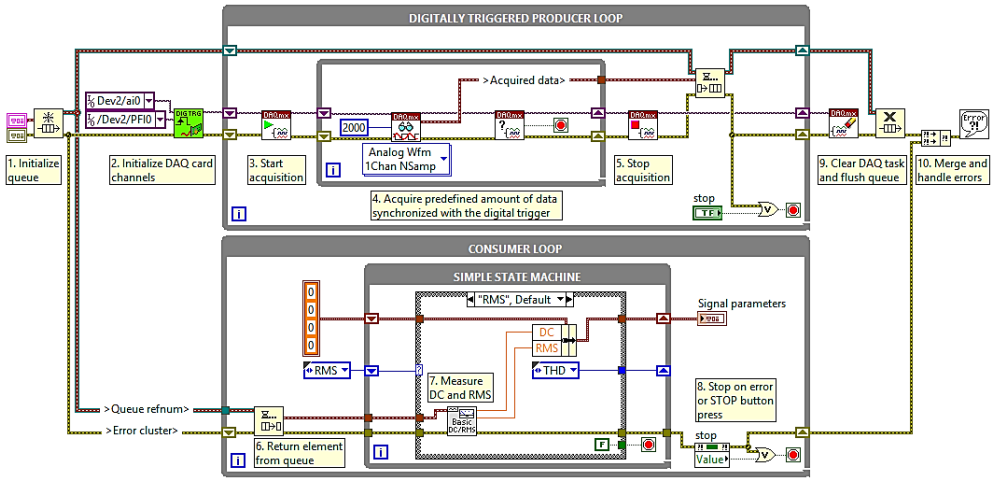
- Дефинирање назив на аналоген канал согласно расположливата картичка за аквизиција на податоци. Се подразбира дека картичката за аквизиција мора да подржува интерфејс на аналогни сигнали.
- Мерно подрачје на аналогниот канал и мерна единица (±5 V во конкретната реализација). Други карактеристични униполарни и биполарни напонски нивоа се 1,25; 2,5 V; 5 V и 10V.
- Врска на влезниот канал (диференцијална или еднокрајна конфигурација). Кај еднокрајната конфигурација едниот крај од влезниот сигнал е поврзан на заземјувањето на системот, додека

кај диференцијалната конфигурација ниту еден од влезните приклучоци не не е заземјен.

По нагодување на карактеристиките на влезниот канал се преминува кон дефинирање на условите за земање одбироци. Суштински параметри се брзината на земање одбироци, број на одбироци и вид на аквизиција (конечна или континуирана). (За разлика од континуираната аквизиција, конечната аквизиција запира по земањето на дефинираниот број примероци). Односот на првите два параметри го одредува времето на аквизиција на податоците. На пример, во примерот прикажан на сликата 2.18 нагодена е брзина 10000 примероци во секунда и вкупно 2000 примероци. Можеме да заклучиме дека, со таква брзина, 2000 примероци ќе се добијат за 1/5 дел секундата (односно за 200 ms). Имајќи предвид дека сигналот на електроенергетската мрежа има фреквенција од 50 Hz, односно периода од 20 ms, така нагодениот период на аквизиција ќе покрие вкупно десет периоди од сигналот. Согласно теоремата за земање одбироци, фреквенцијата на семплирање треба да биде најмалку два пати поголема од максималната фреквенција од сигналот. Тоа значи дека фреквенцијата на семплирање од 10 kHz ќе ги задоволи критериумите за анализа на сигнал со фреквенциски спектар до 3 kHz. Во следниот чекор од програмата се врши нагодување на карактеристиките на сигналот за тригерирање, што претставува една од суштинските фази при синхронизираната аквизиција на податоци. Постојат две основни категории на тригерирање: со аналоген или со дигитален сигнал. При тригерирање со дигитален сигнал потребно е да се нагоди и активниот раб на тригерирање (растечки или опаѓачки) како и дигиталниот канал на кој физички е поврзан сигналот за тригерирање. За да се одговори на барањата на проектната задача во реализацијата прикажана на сликата 2.18 е нагоден растечки раб за тригерирање. До овој момент се нагодени сите неопходни карактеристики на влезниот канал. Аквизицијата започнува и завршува со повикување на соодветната функција за старт/стоп. Помеѓу функциите за започнување и запирање на аквизицијата вообичаено се поставува итеративен циклус со кој се врши аквизицијата на податоците. Аквизицијата започнува со растечкиот раб на дигиталниот тригер, а завршува во моментот кога специјализираната функција за проверка на статусот на задачата ќе врати високо логичко ниво.

Во основа реализацијата прикажана на сликата 2.18 целосно ја опишува синхроната аквизицијата на аналоген сигнал со дигитален тригер. Меѓутоа, во практичните реализации покрај проста аквизиција на сигналот од интерес вообичаено се наметнуваат и други барања како што се: обработка на сигналот, пресметка на одредени параметри, снимање

во датотеки, управување и др. Овие барања наметнуваат примена на посложени програмски архитектури. За да го објасниме ова, ги прошируваме барањата на нашиот пример поврзан со квалитет на електрична енергија на тој начин што дополнително воведуваме барања за пресметка на следните параметри: еднонасочно ниво на сигналот, ефективна вредност, основна фреквенција и вкупно хармонско изобличување. Реализацијата на виртуелниот инструмент со проширените барања е прикажан на сликата 2.19.

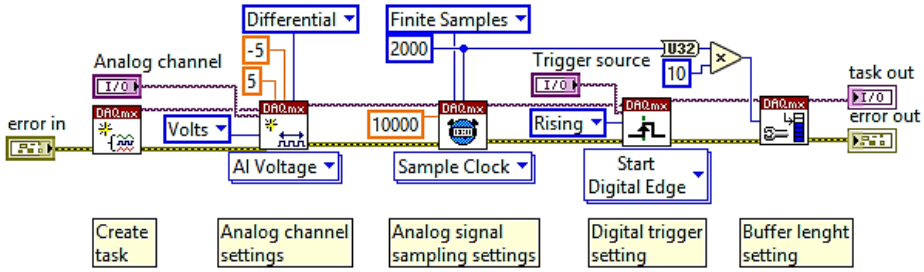


Сл. 2.19 Интегрирање на програмските архитектури производител-корисник и машина на состојби за реализација на виртуелен инструмент за синхрона аквизиција на податоци со дигитален тригер

Основа за реализација на виртуелниот инструмент е програмската архитектура производител-корисник. Програмата е реализирана со два *while* циклуси од кои циклусот во горниот дел од сликата претставува "производител", додека циклусот во долниот дел е "корисник". Циклусот на производителот се користи за синхрона аквизиција на аналогниот сигнал со надворешен дигитален тригер. Од друга страна циклусот на корисникот се користи за пресметка на параметрите согласно дефинираните проектни барања. Основна предност на предложената програмска архитектура е временската ефикасност во аквизицијата на сигналот и пресметка на параметрите. Имено, на овој начин се овозможува паралелна работа на двата циклуси во програмата и на тој начин се избегнува испуштање на делови од аналогниот сигнал за време на пресметките.

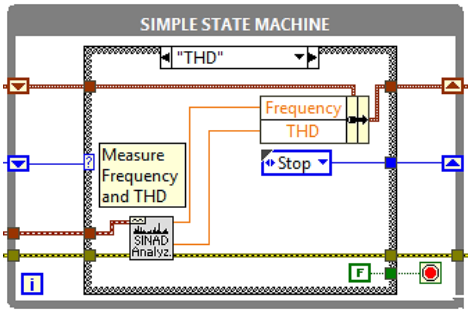
Виртуелниот инструмент започнува со регистрирање и иницијализација на бафер за меморирање на одбироците од влезниот аналоген сигнал. Референцата на баферот истовремено се предава на циклусисте на производителот и корисникот. Нагодувањето на

картичката за аквизиција на податоци се реализира со под-виртуелниот инструмент (нумериран со број 2 на сл. 2.19) кој ги следи принципите елаборирани на сликата 2.18. Приказ на реализацијата на под-виртуелниот инструмент еприкажан на слика 2.20.



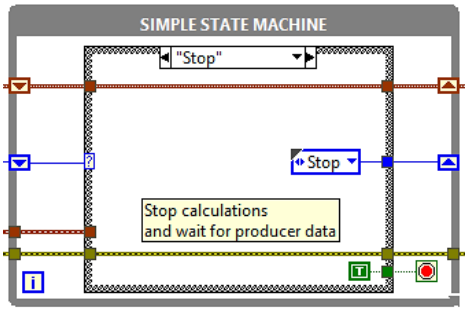
Сл. 2.20 Реализација на под-виртуелен инструмент за синхрона аквизиција на аналоген сигнал со дигитален тригер

Може да се забележи дека во реализацијата на под-виртуелниот инструмент има уште една дополнителна функција со која се нагодува должината на баферот на влезниот канал. Во конкретната реализација, баферот на влезниот канал е нагоден така што може да акумулира десет рамки од влезниот аналоген сигнал. Намената на циклусот на производителот е да генерира рамки од аналогниот сигнал и да го полни баферот. По автоматизам, штом баферот во циклусот на производителот се наполни со податоци, тогаш баферот во циклусот на корисникот ги предава податоците за понатамошна обработка. Во меѓувреме циклусот на произодителот продолжува да акумулира податоци од аналогниот сигнал. За да се овозможи флексибилност на решението и едноставно додавање на нови функционалности, во циклусот на корисникот е имплементирана проста машина на состојби. Првата состојба на машината е пресметка на ефективната вредност и еднонасочното ниво на сигналот RMS. По пресметка на параметрите тие се запишуваат во податочен поместувачки регистар со цел ажурирање на вредностите на кластер индикаторот Signal parameters. Следна состојба во машината на состојби е пресметка на основната фреквенција и вкупното хармонско изобличување на сигналот THD. Исечок од блок-дијаграмот на виртуелниот инструмент со кој се илустрира оваа состојба е прикажан на слика 2.21.



Сл. 2.21 Исечок од машината на состојби и приказ на состојбата за мерење фреквенција и вкупно хармонско изобличување THD

Се забележува дека реализацијтаа е слична како и претходно со таа разлика што се пресметуваат други параметри на сигналот. На овој начин можат едноставно да се додаваат и други состојби во машината на состојби кои можат да извршуваат по комплексни задачи и дијаграми на премин. Пресметките на параметрите на сигналот завршуваат во состојбата *Stop* (сл. 2.22) по што корисникот чека нови податоци од производителот.



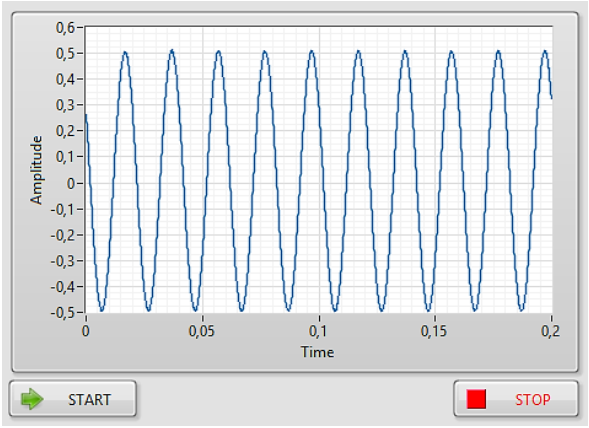
Сл. 2.22 Исечок од машината на состојби и приказ на состојбата за запирање на пресметките *Stop*

Работата на виртуелниот инструмент може да се запре на два начина: со контролното копче *Stop* или при појава на грешки индицирани низ кластерот за грешка. Пренос на вредноста на контролата *Stop* во циклусот на корисникот се врши со примена на јазол за карактеристики. По запирање на двата циклуси на производителот и корисникот се врши празнење на баферот, одјавување на задачата за каналот за аквизиција и приказ на можни грешки.

2.2.2 Тригер од контрола преден панел

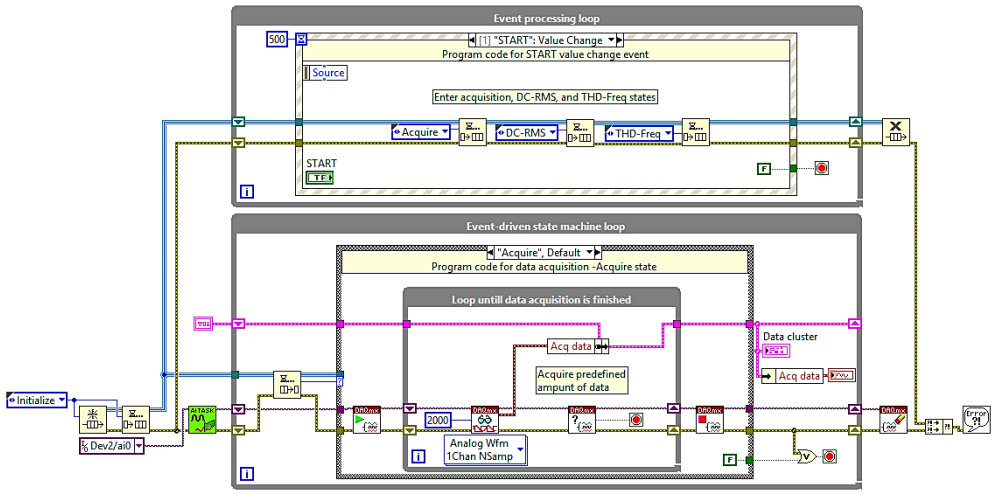
Покрај аквизицијата на податоци синхронизирана со надворешен тригер сигнал, истата може да се реализира и во синхронизација со акција на корисникот или при промена на вредноста на одредена

контрола/индикатор на виртуелниот инструмент. Основната идеа во овој пример е аквизицијата на сигналот да започне со акција на одредена логичка контрола на предниот панел на виртуелниот инструмент. Со самата акција на контролата, виртуелниот инструмент врши аквизиција на десет периоди од аналогниот сигнал, го прикажува сигналот и ги пресметува неговите параметри. Основен изглед на предниот панел на виртуелниот инструмент е прикажан на сликата 2.23.



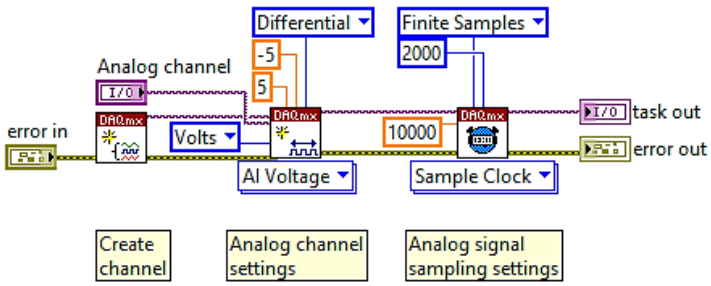
Сл. 2.23 Изглед на предниот панел на виртуелниот инструмент за аквизиција синхронизирана со акција врз контролата *START*

Од примерот прикажан на сликата 2.23 може да се забележи дека почетокот на аквизицијата на сигналот е случајна, односно поместувањето на сигналот во однос на нултиот премин ќе биде исто така случајно. Иако на прв поглед решението на проблемот може да изгледа слично како примерот елаборирaн во претходното поглавје, сепак во овој случај постојат суштински разлики во приодот на реализацијата на проблемот. Имено, наспроти решението од претходниот пример каде што регистрирањето на тригерот се реализира на хардверско ниво, во овој случај настанот се реализира на софтверско ниво со употреба на структури за регистрирање прекин. Едно можно решение кое нуди добра флексибилност во поглед на додавање нови функционалности на системот во иднина е користење на архиетктура машина на состојби која е управувана од структура за прекин. Предложеното решение е прикажано на сликата 2.24.



Сл. 2.24 Архитектура машина на состојби управувана од структура за прекин за аквизиција синхронизирана со акција на контрола/индикатор на виртуелниот инструмент

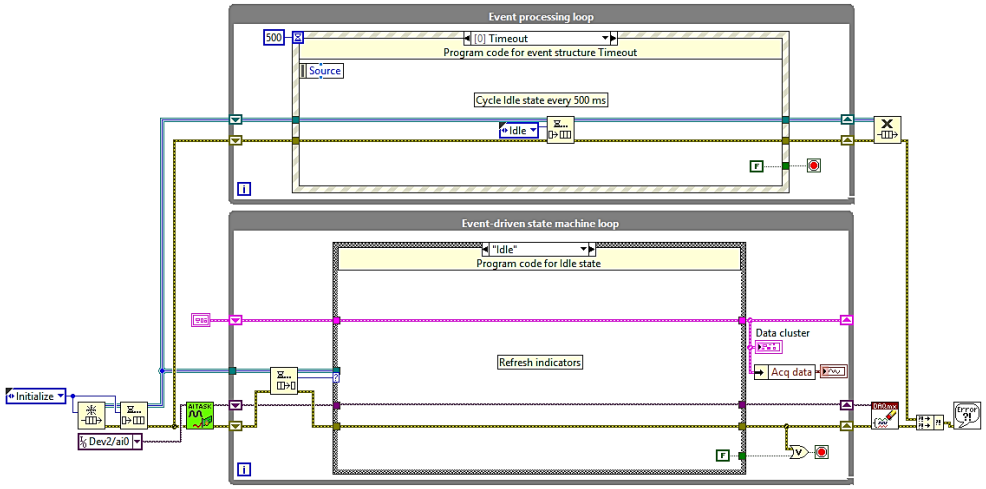
Реализација на дијаграмот на премин на машината на состојби се реализира со употреба на бафери. За таа цел, искористена е типски дефинирана енумерирана контрола со која се регистрира и иницијализира баферот. Првиот податок кој е меморираn во баферот е состојбата *initialize* што значи дека тоа ќе биде и првата состојба која ќе се изврши во машината на состојби. Референцата на баферот и во овој случај се предава на двата *while* циклуси затоа што управувањето со податоците во баферот се реализира и во структурата за прекин и во машината на состојби. По регистрирање на баферот, програмата врши нагодување на карактеристиките на каналот за аквизиција со помош на под-виртуелен инструмент. Изглед на реализираниот под-виртуелен инструмент е прикажан на сликата 2.25.



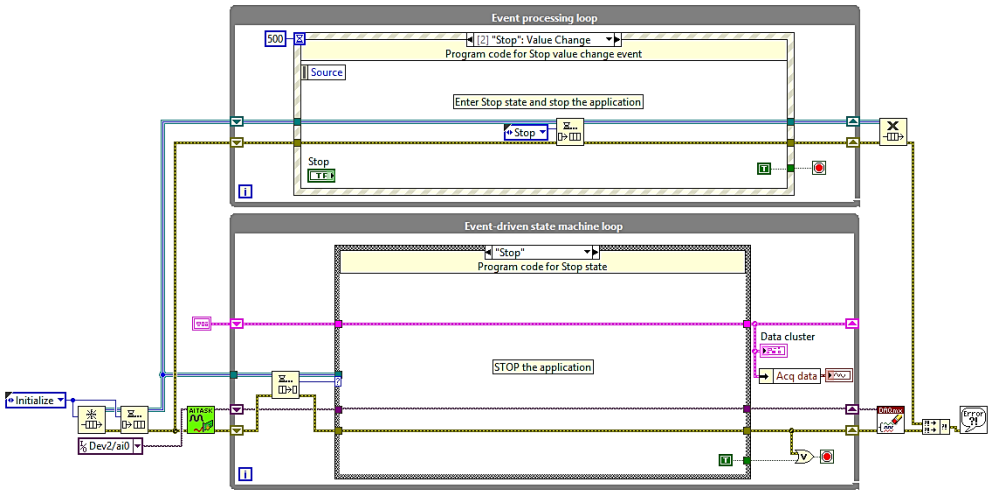
Сл. 2.25 Изглед на под-виртуелен инструмент за нагодување на аналоген канал за аквизицијана податоци

Може да се забележи дека во споредба со под-врuелниот инструмент прикажан на сликата 2.20, во овој случај не се користи функцијата за нагодување на хардверски тригер. Останатите нагодувања на влезниот

канал во основа остануваат исти. Структурата за прекин може да регистрира три вида настани: изминување на времето, промена на вредноста на контролата *Start* и промена на вредноста на контролата *Stop*. *Timeout* настанот на структурата за прекин се активира на секои 500 ms и во тој случај во баферот се додава состојбата *Idle*. Оваа состојба служи само за повремено освежување на програмата и контролите/индикаторите на виртуелниот инструмент. Слично, прекинот Stop се регистрира со притискање на контролата *Stop* и служи за запирање на двата циклуси од виртуелниот инструмент. Приказ на *Timeout*/*Idle* и *Stop*/*Stop* состојбите се прикажани соодветно на сликите 2.26 и 2.27.

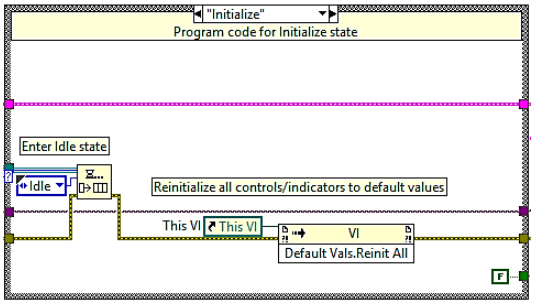


Сл. 2.26 Приказ на *Timeout* настанот на структурата на прекин и *Idle* состојбата на машината на состојби



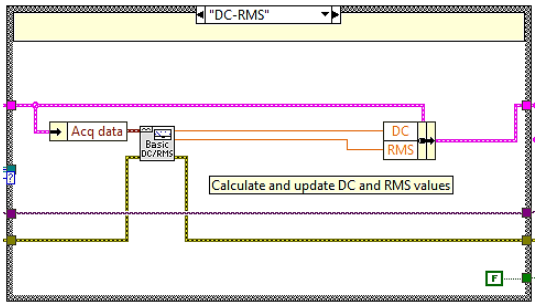
Сл. 2.27 Приказ на *Stop* настанот на структурата на прекин и *Stop* состојбата на машината на состојби

Запирањето на циклусите може да се случи и доколку од кои било причини се појави грешка во кластерот за грешки. Аквизицијата на аналогниот сигнал се врши со регистрирање на прекин со притискање на контролата *START*. Од реализацијата на виртуелниот инструмент прикажан на сликата 8 се забележува дека по регистрирање на настанот во баферот се додаваат три состојби кои се извршуваат редоследно: *Acquire*, *DC-RMS* и *THD-Freq*. Приказ на исечок од останатите состојби во машината на состојби се прикажани на сликите 2.28, 2.29 и 2.30.

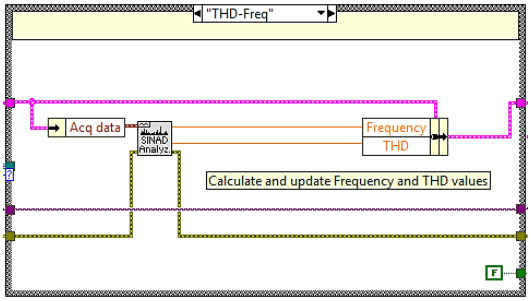


Сл. 2.28 Приказ на состојбата *Initialize* во машината на состојби

Во состојбата *Initialize* со помош на јазол за карактеристики се врши реиницијализација на контролите и индикаторите на основните вредности. Покрај тоа, во баферот се додава состојбата *Idle*. Во основа, додавањето на оваа состојба е непотребно бидејќи истото се постигнува и при активирање на *Timeout* настанот, меѓутоа во примерот се анализира ваквата реализација како можност за додавање на состојби и од страна на машината на состојби.



Сл. 2.29 Приказ на состојбата *DC-RMS* во машината на состојби



Сл. 2.30 Приказ на состојбата *THD-Freq* во машината на состојби

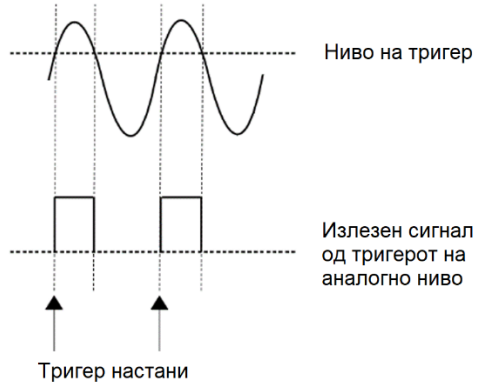
Во состојбата *Acquire*, сналогниот сигнал се снима и се меморира во кластерот за податоци за понатамошна обработка. Потоа, состојбите *DC-RMS* и *THD-Freq* се врши пресметка на параметрите на сигналот (еднонасочно ниво, ефентивна вредност, основна фреквенција и вкупно хармонско изобличување). На сличен начин, преку структирата за прекин, или пак во машината на состојби можат да се додаваат нови состојби со кои ќе се внесат нови функционалности и разгранување на програмата. По правило, на излезот од двата циклуси, програмата врши одјава на баферот и каналот за аквизиција како и спојување и приказ на можни грешки.

2.2.3 Тригер од аналоген сигнал

Аквизицијата на податоци може да се изврши и со примена на аналоген тригер. Предноста во однос на тригерирањето со дигитален сигнал е во тоа што во овој случај се користи само еден влезен канал од картичката за аквизиција на податоци. Ова е овозможено на тој начин што тригерот се дефинира преку одредени амплитудни карактеристики на аналогниот сигнал. Може да се дефинираат три вида тригери: тригер на ниво, тригер на ниво со хистерезис и тригер со прозорец.

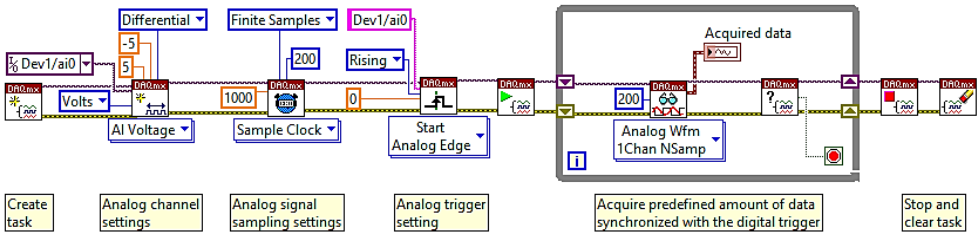
2.2.4 Тригер на ниво

Тригер на ниво на аналоген сигнал се генерира во случај кога аналогниот сигнал ќе надмине одредено дефинирано амплитудно ниво. Тригерирањето на ниво е илустрирано на сл. 2.31.



Сл. 2.31 Тригерирање на ниво на аналоген сигнал. Стрелките означуваат тригерирање на растечки раб на сигналот.

Аналогниот влезен канал содржи компаратор кој го споредува аналогниот сигнал со референтното ниво на тригерирање. Доколку сигналот го надмине референтното ниво, компараторот генерира високо логичко ниво. На тој начин, на излезот од компараторот се добива дигитален сигнал (прикажан во долниот дел на сл. 2.31) кој може да се искористи за генерирање на тригер на растечкиот или опаѓачкиот раб на сигналот. Конфигурирањето на тригерирањето на ниво на аналоген сигнал во виртуелнен инструмент е прикажано на сликата 2.32.



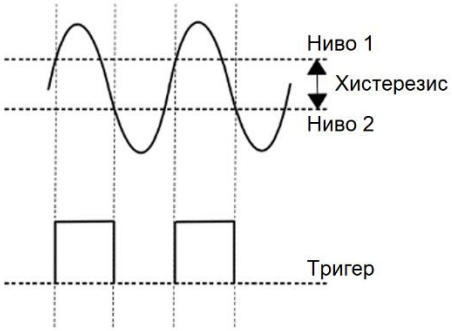
Сл. 2.32 Реализација на виртуелен инструмент за аквизиција на англоген сигнал преку тригерирање на ниво

Од примерот прикажан на сл. 2.32 може да се забележи дека реализацијата не се разликува премногу од таа прикажана на сл. 2.18. Виртуелниот инструмент се разликува единствено во функцијата за дефинирање на тригерот. Во овој случај се избира тригер на аналоген раб на сигналот, при што неопходно е дополнително да се нагоди нивото на тригерирање (во случајот 0 V) и активниот раб на тригерирање (во случајот растечки раб). Покрај тоа се дефинира и каналот на аналогниот сигнал кој во овој случај функцијата го прима во вид на податочен тип *string*. Со оглед на тоа што аналогниот канал во функцијата за нагодување на каналот е ист со каналот кај функцијата за тригерирање, имплементацијата може да се подобри со конверзија на називот на

влезниот канал во податочен тип string и поврзување со функцијата за тригерирање. Со овие измени реализацијата прикажана на сл. 2.32 може директно да се примени во архитектурата производител-корисник дадена на слика 2.19. Како што спомнавме, таквата реализација би имала предност бидејќи користи помалку влезни канали за реализација со иста функционалност. Меѓутоа, тоа неможе секогаш да се примени бидејќи сите картички за аквизиција не подржуваат тригерирање со аналоген сигнал.

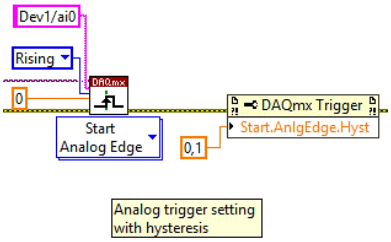
2.2.5 Тригер на ниво со хистерезис

Аналогниот сигнал е секогаш опфатен од влијанија на шум. Во таков случај тригерирањето на ниво на аналоген сигнал може да биде проблематично бидејќи може да се појават неколку последователни тригери во моментот на премин на нивото на тригерот. Ова може да се надмине доколку се примери тригер на ниво со хистерезисна функција илустриран на сликата 2.33.



Сл. 2.33 Тригер на ниво на аналоген сигнал со хистерезисна функција

При ваквата реализација постојат два прага на премин: горен и долен праг (на сликата означени со ниво 1 и ниво 2). Дигиталниот сигнал за тригерирање добива високо логичко ниво кога аналогниот сигнал ќе гопремине горниот праг, а ниско ниво кога истиот ќе го премине долниот праг (или обратно). На таков начин се спречува можноста за лажно тригерирање. Реализацијата на виртуелниот инструмент повторно се разликува само во функцијата за тригерирање која треба да биде нагодена како на сликата 2.34.

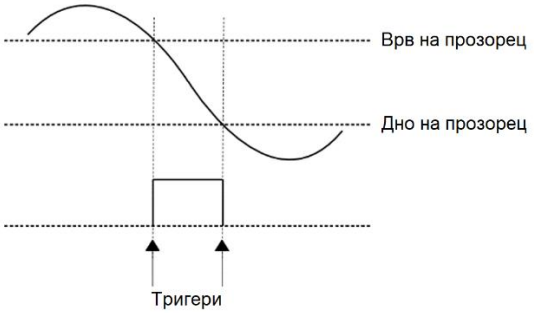


Сл. 2.34 Нагодување на функцијата за тригерирање за примена на тригер на ниво на аналоген сигнал со хистерезисна карактеристика

Во реализацијата прикажана на сликата 2.34 е нагодена хистерезисна функција со растечки раб каде што долниот праг на премин е 0 V, додека горниот праг изнесува 0,1 V. Јасно е дека нивото на праговите зависат од распонот на аналогниот сигнал и од нивото на влијанието на шум.

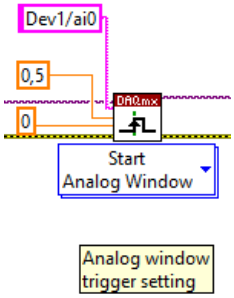
2.2.6 Тригер со прозорец

Тригерирањето со аналоген сигнал може да биде реализирано и со помош на прозорец прикажан како на сликата 2.35.



Сл. 2.35 Тригер со прозорец на аналоген сигнал

Слично како и кај хистерезисната функција, тригерот со прозорец се карактеризира со два прага на премин на сликата означени со “врв на прозорец” и “дно на прозорец”. Тригерот може да се генерира при растечки или опаѓачки раб на аналогниот сигнал при последователен премин на двата дефинирани прага на премин. Овој пристап често се користи при анализа на апериодични сигнали или сигнали опфатени со влијание на шум. Нагодувањето на функцијата за тригерирање кај тригер со прозорец е прикажано на сликата 2.36.



Сл. 2.36 Нагодување на функцијата за тригерирање кај тригер со прозорец на аналоген сигнал

Покрај нагодување на функцијата во режим за тригерирање со прозорец, во овој случај потребно е да се нагодат вредностите на горниот и долниот праг на премин (во конкретната реализација нагодени на 0V и 0,5 V).

3. Мрежна виртуелизација

Мрежна виртуелизација (на англ. *Network Virtualization*, NV) е процес на комбинирање на хардверски и софтверски мрежни ресурси и мрежна функционалност во единствен административен ентитет базиран на софтвер, наречен виртуелна мрежна рамка.

Software-defined networking (SDN) и *Network Function Virtualization* (NFV) практично се нови пристапи кон дизајнот на мрежи, распоредувањето и функционирањето на комуникациските мрежи, кои заедно се нарекува виртуализација на мрежи. SDN ја одделува контролата на мрежата од функцијата за пренасочување, ја централизира мрежната контрола и го прави однесувањето на мрежните јазли и уреди да можат да се програмираат. Самата NFV користи технологија за виртуелизација на ИТ, одвојува мрежни функции од доделени, често сопственички хардверски уреди. NFV може да се имплементира без SDN, и SDN исто така не бара виртуелизација на мрежа.

Во оваа поглава се дискутира зошто овие технологии се жешки теми во науката, истражувањето и воопшто зошто толку брзо се прифаќаат на пазарот. Воедно ќе стане збор и за тоа како придобивките на SDN и NFV се применуваат на 5G мрежите, сателитските мрежи и во паметните IoT мрежи.

Најпрво, започнуваме со сумирање на техничките можности на SDN и дискутирање за неговите придобивки. Следно, ги разгледуваме карактеристиките и придобивките на NFV. На крајот даваме соодветна споредба на овие две технологии и заклучоци за истите. Не треба да се заборави дека, трендот надвор од софтверски дефинираните инструменти им дава на инженерите беспрекорна контрола врз автоматизираните тест системи и овозможувајќи нови видови на сервиси поддржани со мрежната виртуализација (SDN и NFV).

3.1 SDN

Управувањето со мрежи се соочува со нови предизвици, или во мрежни политики (управување со права, *Access Control List*-и (ACL) или изолација меѓу станарите), безбедност (нови закани) или дури и во управувањето со сообраќајот (*Quality of Service*). Во исто време, мрежите стануваат се подинамични поради Cloud Computing (Пресметки во облак) или средните кутии (*Firewalls*, IDS) што ги бараат мрежните администратори.

Конвенционалните мрежи користат посебни алгоритми на посветени уреди (хардверски компоненти) за контрола и следење на протокот на

податоци во мрежата, управување со патеки за рутирање и одредување на тоа како различни уреди се меѓусебно поврзани во мрежата. Општо земено, овие рутирани алгоритми и множества на правила се имплементираат во посветени хардверски компоненти. Сепак, мрежите заостануваат и користат сè уште стари слоеви технологии со интерфејси специфични за продавачите. Така, таа е комплицирана и склона кон грешки за управување со мрежата.

Во конвенционална мрежа, по приемот на IP пакет во уред за рутирање, се користат збир на правила (вградени во рутерот) за да се пронајде дестинацискиот уред, како и патот за рутирање за тој пакет. Обично со сите пакети со податоци што би требало да бидат доставени до истата дестинаци се постапува на сличен начин. Оваа операција се одвива во ефтини уреди за рутирање. Поскапите уреди за рутирање можат да ги третираат различните видови пакети на различни начини врз основа на нивната природа и содржина. На пример, одредени рутерот им овозможуваат на корисниците да ги одбележат приоритетите на различните сообраќајни потоци преку прилагодено локално рутерско програмирање. Така може директно да се управува со големините на редовите на чекање во секој рутер. Таков еден прилагоден локален рутер со можности за подесувања овозможува поефикасна контрола на сообраќајот и контрола на приоритетите, како и севкупен зголемен квалитет на сервиост од крај до крај.

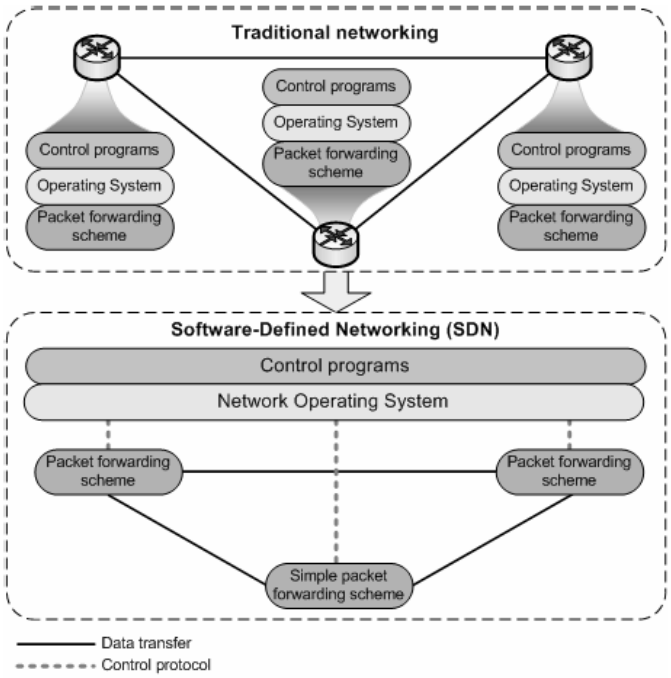
Проблемот што го поставува оваа методологија е ограничувањето на сегашните мрежни уреди под голем мрежен сообраќај, што претставува сериозна ограниченост за мрежните перформанси. Прашањата како што се зголемената побарувачка за приспособливост, сигурност, сигурност и брзина на мрежа, може сериозно да ги попречат перформансите на постојните мрежни уреди поради зголемениот мрежен сообраќај и особено поради развојот на Интернетот. Можно решение за овој проблем е имплементацијата на правилата за управување со податоците и рутирањето, како софтверски модули, наместо да се вградуваат во хардверот. Овој метод им овозможува на мрежните администратори да имаат поголема контрола над мрежниот сообраќај и затоа имаат голем потенцијал за значително подобрување на перформансите на мрежата во смисла на ефикасно користење на мрежните ресурси. Таквата идеја е дефинирана во иновативна технологија, наречена Софтверски дефинирано вмрежување (SDN), што е всушност опфатено во оваа глава.

Неговиот концепт првично бил предложен од страна на Nicira Networks врз основа на нивниот претходен развој на UCB, Stanford, CMU, Princeton [1]. Софтверско дефинирано мрежно поврзување (SDN) е растечка мрежна архитектура што се појавува како жешка тема денес,

со што се разграничува мрежната контрола и функциите за рутирање, овозможувајќи мрежната контрола да стане директно програмабилна и основната носечка инфраструктура да може да се апстрахира за апликациите и мрежните сервиси. Со тоа добиваме централизирана команда и контрола во мрежата, одвојува ја контролната рамнина од рамнината на податоци и рамнината за управување.

Така, може да се рече дека целта на SDN е да обезбеди отворен, контролиран од страна на корисникот управување со хардвер за препраќање и рутирање во мрежа.

Мора да кажеме дека концептот на SDN не е сосема нов. Всушност, пред неколку децении луѓето можеа да користат специјална инфраструктура (како облак-базиран компјутерски хардвер) за да го одвојат мрежниот оперативен систем (сличен на контролните функции во контролната рамнина на SDN) од компјутерски-интензивни апликации (слично на испораката на податоци во рамнина на податоци). Врз основа на идејата за централизирано управување, SDN го адресира овој проблем со раздвојување на рамнината на податоци и контролната рамнина. Тоа е различно од традиционалното IP мрежно поврзување, како што е истото илустрирано на сл. 3.1.



Сл. 3.1. Споредба на концептот на традиционалното омрежување (горе) и SDN (долу)

Принципот на SDN е едноставен: контролниот слој ќе ги реши тековните мрежни барања како што е наведено од страна на мрежниот администратор, користејќи најнови и ефикасни технологии и ќе ги пренесува основните пресметани правила преку податочниот слој, кој може да биде, на пример, едноставни комутатори со стандарден интерфејс како протоколот *OpenFlow*. Овој протокол му овозможува на контролниот слој да ги специфицира правилата (од OSI нивото 2-4: Ethernet, IP, TCP/UDP) за тоа како комутаторот ќе мора да се справи со пакетите. Затоа, контролната рамнина ќе може да користи ресурси на високо ниво (како бази на податоци или LDAP), да работи на стандардни сервери (или виртуелни машини) и да биде лесно модифицирана, обезбедувајќи флексибилност и приспособливост, додека перформансите се гарантираат со употреба на посветен хардвер за комутаторите. Како последица, SDN ги адресира сегашните очекувања со обезбедување на флексибилна и програмабилна мрежа која може да прима информации од комутаторите, корисниците, администраторите и безбедносните уреди и да ги користи за да ја ажурира тековните полиси, во согласност со дефинираните правила кои сега можат да користат софистицирани функции.

Оваа идеја е комплементарна со растечката важност на виртуелизација. Штом сѐ виртуелизира, има смисла да се обиде да ги виртуелизира мрежните јазли и да донесе леснотија за управување. Ова е целта на виртуелизацијата на мрежни функции (NFV). Тоа е дизајнирано да ги забрза иновациите, бидејќи тестовите и распоредувањето ќе бидат полесни и ќе ја зголемат отвореноста, бидејќи таквите уреди би биле софтверски дизајнирани. NFV е комплементарен со SDN, бидејќи постојните виртуелизирани ресурси може да се користат и за контролната рамнина (процесорот, за складирање) и податочната рамнина (виртуелни комутатори). Покрај тоа, со повторно користење на постојните хардверски и одвоени барања, може да се направат значителни заштеди.

Денес пресметките во облак овозможуваат мрежно пресметување и складирање без користење на локални ресурси. Ваквото одвојување на контролата и податоците игра клучна улога во голем систем со големи брзини. SDN резултира со подобрени мрежни перформанси во насока на: управување со мрежата, контрола и обработка на податоци. На тој начин, SDN е потенцијалното решение за надминување на проблемите со кои се соочуваат конвенционалните постојни мрежи (сл.3.1) и добива поголемо прифаќање во апликациите кои се користат во облак. Од перспектива на облак и пресметки во облак, SDN обезбедува одлични придобивки. Прво, тоа го прави облак провајдерот со полесно распоредувањето на различни

производители на уреди. Традиционално големите облак провајдери (како што се Google, Amazon итн.), мора да ги купат високо-квалитетните комутатори/рутери од истиот добавувач, со цел лесно да ги конфигурираат параметрите за рутирање. Различни вендори на рутери имаат свои добри и лоши страни. Сепак, тоа е главоболка при прилагодувањето на секој рутер, бидејќи секој вендор на опрема може да има своја синтакса на јазик. Сега SDN им овозможува на провајдерот на облак брзо да ги преиспита проблемите со рутирање или дистрибуција на ресурси се додека секој рутер на вендорот го следи SDN стандардот. Второ, тој им овозможува на корисниците на облак поефикасно да ги користат ресурсите на облакот или да се спроведуваат научни експерименти со создавање на виртуелни протоци и проточни делови.

Протоколот *OpenFlow*, кој се користи од страна на SDN, е компатибилен со GENI стандардот, и ова му овозможува на корисникот произволно да создаде парчиња, без да биде свесен за физичката мрежна инфраструктура. Без оглед на тоа дали инфраструктурата е безжичен или жичен систем, и без оглед на тоа како облак провајдерот ги распоредува различните единици за складирање на различни локации, концептот на виртуелен проток во SDN прави проток на податоци транспарентно низ сите облак уреди.

Според погоре-кажаното, може да се каже дека SDN воведува нови решенија и техники за виртуализација во мрежите. Виртуализација во SDN може да се воспостави на две места и тоа помеѓу податочната и контролната рамнина или помеѓу контролерот и мрежните апликации.

SDN воведува и виртуелизација на мрежи која има за цел да нуди:

- Независност од мрежната топологија
- Независност од адресниот простор
- Истовремена конфигурација и на процесирачките и на мрежните ресурси
- Можност за комплетно произволна миграција на виртуелни машини
- Автоматско провизионирање низ целата мрежа
- Комплетна изолација на виртуелните мрежи од другите виртуелни мрежи и од самата физичка архитектура

SDN архитектурата има три различни нивоа:

- Апликациски/сервосно ниво (*Application Layer*): Се состои од апликации на крајниот корисник кои користат SDN комуникациски услуги.

- Контролно ниво (*Control Layer*): Ја обезбедува логично централизираната контрола и функционалност која го надгледува мрежното рутирачко однесување преку отворен интерфејс.
- Инфраструктурно ниво (*Infrastructure Layer*): Се состои од мрежни елементи, односно мрежна опрема која врши рутирање и препраќање на пакетите.

Овие три нивоа се достапни преку отворени апликативни програмски интерфејси (APIs). Контролерот на SDN користи API за т.е. „Northbound“ за да комуницира со апликациите "над" и т.е. „southbound“ API-иа за да комуницира со опремата за мрежно поврзување подолу (во нивоата). Одделувањето на контролната рамнина на мрежата (т.е. контролната логика) од рамнината на податоци (т.е. од мрежните уреди) има три клучни предности, а тие се следните:

- Контролата на мрежата станува логично централизирана. SDN Контролерот може да ги презентира мрежните услуги до Апликациското ниво и може да ги користи услугите за рутирање на пакети и услуги за препраќање на *Infrastructure Layer*.
- Мрежната контрола станува софтверска и разновидна. Мрежните администратори можат брзо да одговорат на променливите потреби, можат да одвојат мрежни ресурси на побарувачката и лесно можат да ги менуваат полисите.
- Дизајнот и функционирањето на мрежата може да биде поедноставно и поефтино, бидејќи платформата за испорака на сервиси не мора да вклучува кориснички уреди и протоколи.

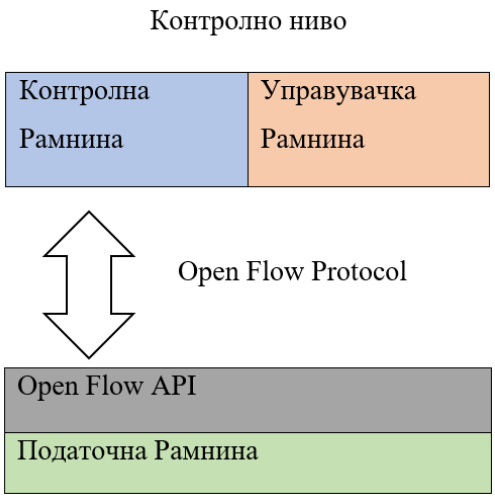
3.1.1 Споредба на SDN и традиционални мрежи

Во овој дел ќе направиме споредба меѓу SDN и традиционалните мрежи, и ќе наведеме разлики помеѓу истите.

Сл. 3.2. Традиционална архитектура

Слика 3.2 ги отсликува слоевите во рамките на традиционален мрежен уред. Тој се состои од контролна рамнина, управувачка рамнина и податочна рамнина. Овие мрежи уште се нарекуваат и статични мрежи. Слика ги отсликува слоевите во софтверско дефинираните мрежи. Се состои еден слој на податоци заедно со *Open Flow* API (отворен проток

API). Ова е поврзано со контролер кој управува со податочната и управувачката рамнина, а над нив се наоѓа апликациското ниво.



Сл. 3.3 SDN архитектура

Табела 3.1. Споредба на традиционално вмрежување и SDN

Традиционално вмрежување	**Software Defined Nerworking**
Ова се статични и нефлексибилни мрежи. Не се корисни за нови деловни потфати. Исто така поседуваат мала агилност и флексибилност.	Ова се програмабилни мрежи за време на развојот, како и во подоцнежната фаза при промена на барањата. Преку повеќе особини, како флексибилност, агилност и виртуелизација, тие им помагаат на новите деловни потфати.
Тие се хардверски апарати.	Конфигурирани се користејќи слободен софтвер.
Имаат дистрибуирана контролна рамнина.	Имаат логички цетрализирана контролна рамнина.
Работат користејќи протоколи.	Користат API за конфигурирање според потребите.

Во традиционалната мрежа имаме дистрибуирана контролна рамнина (види Табела 3.1), додека во SDN, централизирана е контролната рамнина и управува со протокот на податоци. Главната разлика меѓу овие две мрежи е во потокот на податоци во мрежата. SDN е пофлексибилна во споредба со традиционалната мрежа и лесно прави промени во софтверот во контролната рамнина и промената се појавува на целиот систем за вмрежување кога и да е побаран. Но, во традиционалните мрежи, ние можеме рачно да го менуваме секој од рутерите и соодветно да ги конфигурираме свитчевите.

Најзначајната потреба од SDN е да може да се справи со барањата од големи датацентри, што се должи на виртуелизација на серверот обезбедено од SDN.

Некои од насоките кои ја водат потребата од SDN се:

- Промена на сообраќајните шеми: Во текот на минатите години мрежните сообраќајни шеми се менуваат континуирано. Во клиент-сервер архитектурата, најголемиот дел од преносот на податоци и поделба на ресурси зазема место меѓу еден клиент и еден сервер во време, додека во новите апликации имаме голем пристап кон различни бази на податоци и сервери кои создаваат брзина во мрежата и ја отежнуваат брзината на мрежата, и затоа побаруваат поголем опсег.
- Потрошувачка на IT: Денес, најголемиот дел од луѓето користат паметни телефони, таблети, и слично за да добијат пристап на интерет. Светот на IT е под големо задоволство да се присподоби на ваквите мобилни уреди на соодветен начин и исто така ја заштитува интимноста на корисникот како и корпоративните податоци од измама или кражба.

3.1.2 Предности и цели на SDN мрежите

Самите SDN базирани мрежи се поевтини поради универзални пренасочувачки уреди за пренос на податоци што следат одредени стандарди и обезбедуваат поголема контрола врз протокот на мрежниот сообраќај во споредба со конвенционалните мрежни уреди.

Покрај тоа, реакцијата на мрежна модификација (нова интерконекција или нов уред) или мрежен настан (како напад или сомневање за упад) е лесна и ефикасна, бидејќи лесно можеме да ја смениме целата мрежа за да ја преземеме оваа промена во ред. Покрај тоа, ова може да се направи колку што е можно поблиску до изворот и да се избегне бескорисно оптоварување во нашата внатрешна мрежа.

Пред SDN, инженерите треба рачно да ги менуваат правилата што може да доведат до грешки или бавни реакции. Конечно, централизацијата на логиката во таков целосно прилагоден контролер со глобално знаење и висока компјутерска моќ го поедноставува развојот на пософистицирани функции. Оваа способност за програмирање на мрежата е клучен елемент на SDN.

Во таа насока, главните предности на SDN вклучуваат [2-10]:

- Флексибилност: SDN, исто така, носи голема флексибилност во управувањето со мрежата. Лесно е да се пренасочи сообраќајот, да се испитаат одредени текови, да се тестираат нови полиси или на пример, да се откријат неочекувани протоци, бидејќи контролорот ќе ги прилагоди правилата на ниско ниво на одредени барања. Така, SDN е одлична алатка за тестирање на нов рутирање алгоритам или протокол и за поедноставување на мрежна безбедност, како и откривање на малициозен софтвер.
- Унифицирана полиса: Со својот контролер, SDN исто така гарантира обединета и актуелна мрежна полиси: бидејќи контролорот е одговорен за додавање на пресметаните правила во прекинувачите, не постојат ризици дека мрежниот оператор заборавил прекинувач или инсталира некохерентни правила помеѓу уреди. Навистина, операторот само ќе специфицира ново правило и контролорот ќе ја прилагоди конфигурацијата за испраќање кохерентни правила во секој релевантен уред.
- Напредно рутирање: SDN, исто така, може да се користи за управување со информации за рутирање на централизиран начин преку делегирање на рутирање и користење на интерфејсот за контролорот. Управување со рутирање преку SDN може да се користи за фино управување со користењето на линкови и со тоа подобрување на перформансите преку зголемување на просечната употреба на секоја врска без целосно заситување на врската, бидејќи би било контрапродуктивно.
- Управување со *Cloud*: SDN, исто така, овозможува лесно управување со *Cloud* платформа. Навистина, во таква средина, потребно е одвојување на контролната рамнина со која клиентите и системите комуницираат преку својот API и рамнина за препраќање, додека динамиката донесена од SDN адресира облак-специфични проблеми како приспособливост, адаптација, инстанцијации или движења на виртуелни машини и изолација.

- Хардвер поедноставување: SDN има тенденција да користи стандардни и основни технологии за контрола на мрежна опрема, додека компјутерската моќ е потребна само на контролорот. Така, тековните прекинувачи / рутери не треба да обезбедуваат многу функции и можности за одбивање. Бидејќи сите прекинувачи ќе имаат иста цел, ќе има помалку хардверска разлика, бидејќи тие ќе понудат помалку сопственички карактеристики (кои ќе одат во контролорот). Така мрежната опрема ќе стане ниско-тарифни производи кои нудат стандардни интерфејси. Со таков хардвер, исто така би било едноставно да додадете нови уреди - бидејќи тие не се специјализирани - да ги поврзете со мрежата и да им дозволите на контролорот да ги управува соодветно на дефинираната политика. Така, мрежата лесно ќе стане скалабилна штом контролорот ќе може да се приспособи.
- Хибридно распоредување: SDN сега е зрела технологија која се користи во реални мрежи со голем успех и важни остварувања. Бидејќи SDN претпоставува целосно ангажирање на SDN прекинувачи, не може лесно да се распореди во сите мрежи на претпријатија - кои сеуште можат да имаат наследни прекинувачи и не можат да ја надградат целата своја хардверска опрема во едно време - но е возможен преоден развој со цел да ги искористат предностите на SDN можности. За разлика од типичните шеми за распоредување - SDN мрежата е поставена заедно со постојната мрежа или мрежата е одвоена на парчиња, кои ќе се делат меѓу наследството и новиот SDN. Треба да се земе во предвид и фактот дека е тешко да се филтрира пакет според неговото потекло, ние не можеме да дејствуваме на изворот и не можеме да ги примениме глобалните политики.
- Интелигенција и брзина: SDN имаат способност да ја оптимизираат дистрибуцијата на обемот на работа преку моќен механизам од контролната рамнина (и контролерот, како што беше спомнато погоре). Ова резултира со пренос на голема брзина и поефикасно користење на ресурсите во мрежата.
- Лесно управување со мрежата: Администраторите имаат далечински управувач преку мрежата и можат да ги променат карактеристиките на мрежата, како што се услугите и поврзувањето врз основа на обемот на работа. Ова им овозможува на администраторите да имаат поефикасен и брз пристап до модификации на конфигурацијата.

Понатака, кога станува збор за целите на SDN мрежите, истите можат да се групираат во цели кои тежнеат да реализираат:

- Побрз мрежен бизнис циклус: SDN го намалува времето на одговор на бизнис-барањата до мрежните добавувачи, на пример, за зголемување на клиентското задоволство или да го скрати времето на враќање на инвестицијата преку понатамошна автоматизација на мрежните операции;
- Забрзување на иновациите: SDN ги забрзува деловните и / или техничките иновации преку поголема флексибилност на мрежните операции, со што се олеснува испитувањето и истражувањето;
- Забрзана адаптација кон барањата на клиентите: SDN го олеснува сместување на барањата за поврзување на купувачи преку динамичко преговарање за карактеристиките на мрежните услуги и динамична контрола на мрежните ресурси;
- Подобрена достапност на ресурсите и ефикасност на користењето: SDN е наменета за подобрување на достапноста и ефикасноста на мрежните ресурси, особено кога се комбинира со виртуелизација на мрежата, поради воведувањето на високо ниво на автоматизација во целокупните постапки за испорака на услуги и операција, од преговарање за услужниот параметар до исполнување и осигурување;
- Персонализација на мрежни ресурси, вклучувајќи ги и услугите што се свесни за мрежното поврзување: SDN овозможува мрежно прилагодување за мрежните услуги кои имаат различни барања, преку програмирање на операции со мрежни ресурси, вклучувајќи и динамичко спроведување на серија политики (на пр.: планирање на ресурси како функција на бројот на нарачки на клиенти кои треба да се обработуваат со текот на времето, препраќање и рутирање, квалитет на услуга (QoS) и сообраќаен инженеринг, безбедност итн.).

SDN ги обезбедува следните високо-нивовски можности, кои се клучни во самата основна дефиниција за овие мрежи [10]:

- Програмабилност: Однесувањето на мрежните ресурси може да се прилагоди од SDN апликации преку стандардизиран програмски интерфејс за мрежна контрола и функционалност за управување. Корисникот на интерфејс може да се: мрежни провајдери, даватели на услуги и клиенти, вклучувајќи и крајните

корисници. Ова им овозможува на апликациите на SDN да ги автоматизираат операциите на мрежните ресурси според нивните потреби.

- Апстракција на ресурси: Имотот и однесувањето на основната мрежа ресурсите можат да бидат соодветно апстрахирани и разбрани, оркестрирани, контролирани и/или управувани од оние кои ги програмираат, благодарение на релевантни, стандардни информации и модели на податоци. Овие модели обезбедуваат детален, апстрахиран приказ на физички или виртуелизирани мрежни ресурси.

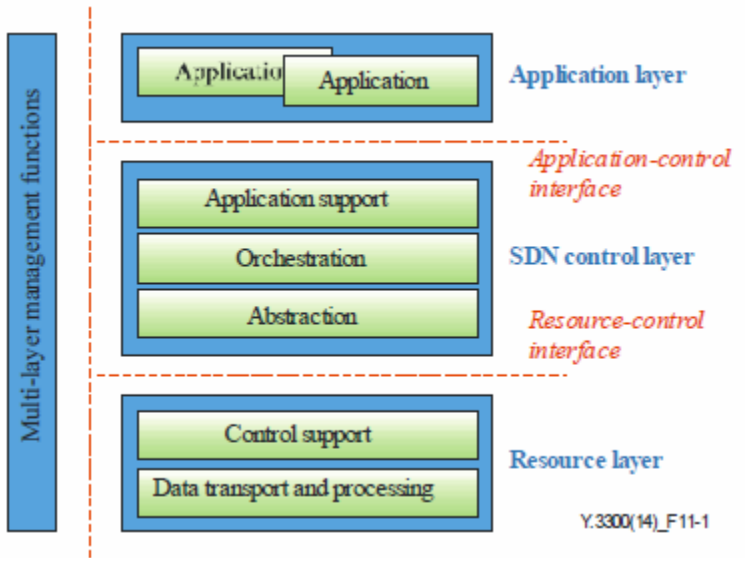
Програмибилност придонесува за воведување на високо ниво на автоматизација во целокупната испорака на услуги, за да се постигне бизнис агилност, како што се креирање динамични услуги и обезбедување. Стандарден интерфејс кој обезбедува канал за интеракции помеѓу SDN апликации и SDN контролери се користи за пристап до информациите од мрежата и за програмирање на апликациско-специфично мрежно однесување. Оваа програмабилност обезбедува можност за контрола или конфигурирање на мрежните елементи преку логички централизиран SDN контролер преку уште еден стандардизиран интерфејс.

Во раните денови на мрежниот инженеринг, само требало да се мачиме со едноставни Ethernet или IP правила и мрежата беше статична, но денес, мрежните барањата стануваат сложени, бидејќи мрежите стануваат се повеќе и подинамични, па затоа се развиваат нови концепти за раздвојување на контролата и податоците – како начин за решавање на овие ограничувања.

Всушност, мрежите мора да бидат одзивни и еластични бидејќи многу компјутери и крајни корисници со разни уреди се поврзани, се преместуваат или се исклучуваат, како преку безжични и мобилни мрежи, така и преку разни сервиси во Облак. Покрај тоа имаме поврзани комплицирани политики, или во безбедноста (на пример, изолација помеѓу соседите, правила на *firewall*-ите, листи за пристап и управување со права) или управување со сообраќајот (квалитет на услуга, балансирање на оптоварување), кои треба да се постават соодветно. Исто така сите мрежи стануваат поголеми и различни мрежи треба да коегзистираат заедно (различни нивоа на безбедност во компаниите, повеќе закупци и корисници во Облак) и имаат индивидуализира мрежа политики.

3.1.3 Архитектура на SDN мрежите

Да се потесетиме дека SDN е збир на техники кои обезбедуваат корисниците директно да програмираат, оркестрираат, контролираат и управуваат мрежни ресурси, кои го олеснуваат дизајнот, доставата и оперирањето со мрежните сервиси на динамичен и скалабилен начин. Во таа насока, високо-нивовската архитектура на SDN се состои од неколку нивоа и истата е прикажана на сл. 3.4.



Сл. 3.4. Високо-нивовска SDN архитектура, [10].

Се состои од неколку слоеви (како што е прикажано на Слика 4), кои ги вклучуваат следново:

- Апликациски слој: таму каде што SDN апликациите ги специфицираат мрежните услуги или деловни апликации со дефинирање на сервисно однесување на мрежните ресурси на програмски начин. Овие апликации комуницираат со SDN контролен слој преку интерфејси за контрола на апликации, со цел за SDN контролен слој автоматски да ги прилагоди однесувањето и својствата на мрежни ресурси. Програмирањето на SDN апликацијата користи апстрахиран поглед на мрежните ресурси, обезбедени од SDN контролниот слој со помош на информации и податоци модели изложени преку интерфејс за контрола на апликацијата.
- SDN контролен слој: Тој обезбедува средства за динамички и детерминистички да може да се контролираат и управуваат

однесувањето на мрежните ресурси (како што се пренос на податоци и преработката), како што е наведено од страна на апликативниот слој. Освен тоа, SDN апликациите наведуваат како мрежните ресурси треба да се контролираат и да се доделени, преку интеракција со SDN контролниот слој преку апликациски-контролирани интерфејси. Нивото на апстракција варира во зависност од апликациите и природата на услугите што треба да се испорачаат. Овој слој вклучува:

- Поддршка за апликации: Функцијата за поддршка на апликацијата која обезбедува интерфејс за контрола на апликации за SDN апликациите да можат да имаат пристап до мрежните информации и програмски специфични мрежни однесувања на апликациите.
- Оркестрација: Функцијата за оркестрација обезбедува автоматска контрола и управување со мрежни ресурси и координација на барањата од слојот за апликација за мрежни ресурси врз основа на политиката обезбедена од повеќеслојните функции за управување или од апликацискиот слој. Функцијата за оркестрација обезбедува контрола и управување со мрежни ресурси, што на пример опфаќа: управување со физички и виртуелни мрежни топологии, мрежни елементи и сообраќај.
- Апстракција: Се поврзува со мрежните ресурси и обезбедува апстракција на мрежните ресурси, вклучувајќи мрежни способности и карактеристики, со цел да се поддржи управувањето и оркестрација на физички и виртуелни мрежни ресурси.
- Ресурсен слој: мрежните елементи го изведуваат транспортот и обработка на пакети со податоци во согласност со одлуките донесени од SDN контролниот слој, и кои се пренасочени кон ресурсниот слој преку интерфејс за контрола на ресурсите.
- Контрола за поддршка: Функцијата за контрола на поддршка се поврзува со SDN контролниот слој и ја поддржува програмабилноста преку ресурсно-контролните интерфејси.
- Пренос и обработка на податоци: Функцијата за пренос и обработка на податоците обезбедува пренасочување на податоци и функционалности за рутирање на податоците.

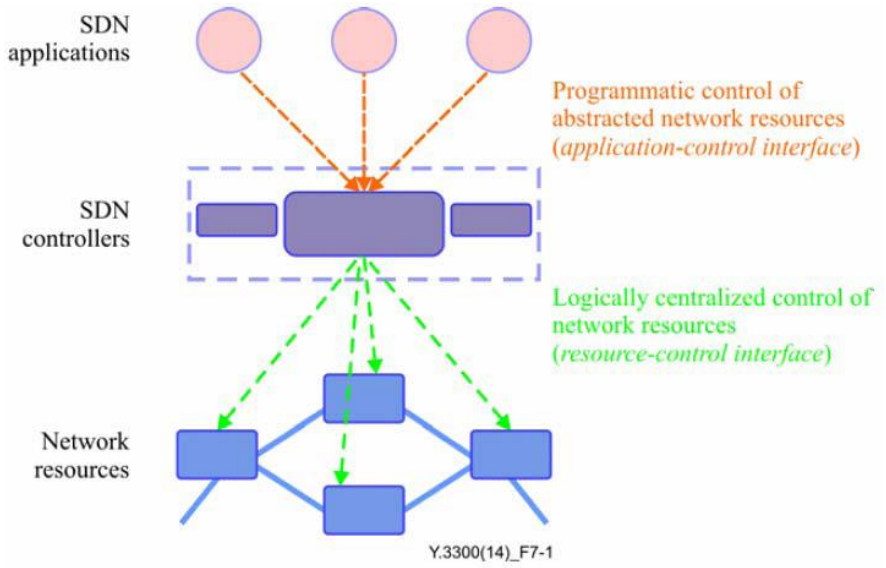
Контролната рамнина е систем кој донесува одлуки каде сообраќајот треба да се испрати, а податочната рамнина е систем за кој се грижи проследување на сообраќајот.

Улогата на рамнината на податоците е едноставна: со оглед на пакет, таа треба да ја пренесе следниот хоп следејќи неколку едноставни - и ако

е можно, генерички - правила (од слој 2 до слој 4 во OSI моделот: Ethernet, IP, TCP и UDP). Така, можеме да обезбедиме перформанси - критична точка во мрежите, бидејќи тие треба да се справат брзо со големи количини на податоци, без важни одложувања или пакети загуби. Контролната рамнина сега може само да се грижи за пренесување на овие правила на ниско ниво кои се пресметуваат од унифицирани и комплексни политики на високо ниво. Со тоа, контролната рамнина може да користи ресурси од повисоките нива како: напредно управување со права со бази на податоци и користење на постоечките ресурси со работа на стандардни сервери или виртуелни машини.

Контролерот исто така може да ги прилагоди овие правила според динамичните потреби, нови политики, мрежни настани или сигнали. Со цел да бидат информирани за вакви настани, на контролерот ќе се изложи API.

SDN ја преместува контролата на мрежните ресурси во посветен мрежен елемент, имено SDN контролер, кој обезбедува средства за програмирање, оркестрирање, контрола и управување со мрежните ресурси преку софтвер (т.е., SDN апликации), како што е прикажано на сл. 3.5. Распределените мрежни ресурси ги извршуваат мрежните функции како што се пренос и обработка на податоци, но однесувањето на мрежните ресурси се контролира директно преку стандардизиран интерфејс (односно интерфејс за контрола на ресурсите) и релевантни информации, заедно со податочни модели.

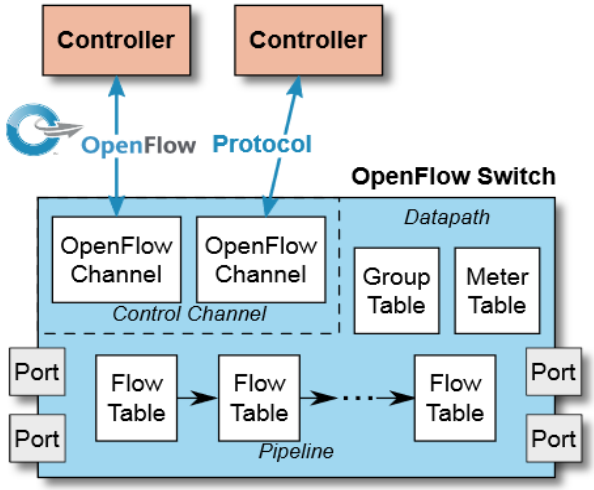


Сл. 3.5. Преглед на SDN, [10].

Контролерот на SDN ги управува и конфигурира дистрибуираните мрежни ресурси и обезбедува апстрахиран приказ на мрежните ресурси на SDN-апликациите преку друг стандардизиран интерфејс (односно интерфејс за контрола на апликацијата) и соодветните модели за информации и податоци. Самата SDN апликација може да ги прилагодува и автоматизира операциите (вклучувајќи управување) на апстрахираните мрежни ресурси на програмабилен начин преку овој интерфејс. Како што може да се забележи, SDN контролерот може да обезбеди различни типови на интерфејси за SDN апликациите (на пример, повеќе апстрахирани и повеќе објектно-ориентирани).

3.1.4 OpenFlow протокол

Постојат голем број стандарди за протоколи за користење на SDN во реални апликации и сервиси. Еден од најпопуларните протоколи се нарекува OpenFlow [11-15]. OpenFlow е протокол кој овозможува имплементација на концептот SDN и во хардверот и во софтверот. Важна карактеристика на OpenFlow е дека научниците можат да го искористат постоечкиот хардвер за да дизајнираат нови протоколи и да ги анализираат нивните перформанси. Овој протокол станал дел од комерцијално достапните рутери и комутатори. Како стандарден SDN протокол, *OpenFlow* бил предложен од Стендфорд. Во врска со тестовите на OpenFlow, биле предложени многу дизајни за *OpenFlow* протоколите. Тие користат кодови со отворен код за контрола на универзалните SDN контролери и комутатори. Во врска со комутаторите, *OpenVSwitch* (OVS) бил еден од најпопуларните софтверски управувани *OpenFlow* комутатори. Неговиот кернал е напишан во Линукс 3,3, а исто така е достапен и неговиот *firmware* вклучува Pica8 [16] и Indigo [17]. OpenFlow е протокол ориентиран кон проток и има абстракции на комутатори и порти за контрола на протокот. Во SDN, постои софтвер наречен контролер кој управува со колекцијата на комутатори за контрола на сообраќајот. Контролаторот комуницира со OpenFlow комутаторот и управува со комутаторот преку OpenFlow протоколот. Самиот *OpenFlow* комутатор може да има повеќе проточни табели, групна табела, и *OpenFlow* канали (види Слика 6 [11]). Каналите остваруваат врски со *OpenFlow* контролероте или со други контролери. Секоја табела за проток содржи записи за протокот и комуницира со контролорот, и групната табела може да ги конфигурира записите за протоците. *OpenFlow* комутаторот се поврзуваат едни со други преку *OpenFlow* портите.



Сл. 3.6. OpenFlow комутатор, [11].

Користејќи го *OpenFlow* протоколот, контролорот може да додава, ажурира и да ги брише записите во поточните табели (т.е. *flow tables*), и реактивно (како одговор на пакети) и проактивно. Секоја табела за поток во комутаторот содржи множество на влезни записи; секој влез на проток се состои од полиња за совпаѓање, бројачи и множество инструкции за прилагодување на пакетите. Почнувањето започнува на првата табела за проток и може да продолжи со дополнителни табели за проток на линијата за процесирање (види 5.1.1 [11]). Евиденциите на протокот се совпаѓаат со пакетите во редослед, со првиот приказ за појавување во секоја табела која се користи. Ако е пронајден соодветен запис, инструкциите поврзани со специфичниот запис за проток се извршуваат (видете 5.5 [11]). Ако во табела so протоци не се пронајде совпаѓање, исходот зависи од конфигурацијата на записот во табелата за проток: на пример, пакетот може да се препрати до контролорите преку *OpenFlow* каналот, да биде отфрлен или може да продолжи во следната табела на проток.

Првично, патеката за податоци на *OpenFlow* рутирање уреди има празна рутирачка табела со некои полиња (како изворна IP адреса, тип QoS, итн.). Оваа табела содржи неколку пакети полиња како што е дестинацијата на различни порти (примање или пренос), како и поле за акција што го содржи кодот за различни дејства, како што се: препраќање на пакети или прием, итн. Оваа табела може да биде населена врз основа на дојдовни податоци пакети. Кога ќе примите нов пакет, кој нема соодветен внес во табелата за проток на податоци, тој се проследува на контролорот што треба да се обработува.

SDN има способност за програмирање на повеќе комутатори истовремено; но сеуште е дистрибуиран систем и, според тоа, страда од конвенционални комплексности како што се намалувањето пакети, одложување на контролните пакети итн. Тековните платформи за SDN (како што се NOX и *Beacon*), овозможуваат програмирање; но сè уште е тешко да ги програмираме на ниско ниво. Со новите протоколи (како OpenFlow), кои се повеќе стануваат дел од стандардната индустрија, SDN станува полесен за имплементација. Контролната рамнина генерира рутирачка табела, додека податочната рамнина – ја оптимизира табелата за да се утврди каде треба да се испратат пакетите [20]. Многу компании користат OpenFlow протоколи во нивните податочни центри и јадрени мрежи за поедноставување на операциите. OpenFlow и SDN овозможи податочните центри и истражувачите полесно да ги апстрахираат и управуваат големите и сложени мрежи.

Архитектурата на *OpenFlow* обично ги вклучува следните 3 важни компоненти [11], [21]:

1. Комутатори: *OpenFlow* дефинира протокол со отворен код за следење / менување на табелите за протокот во различни комутатори и рутери. Комутаторот *OpenFlow* има најмалку три компоненти:

   а) табели за проток, секоја со поле за дејство поврзано со секој влез на проток,

   б) комуникациски канал, којшто обезбедува линк за пренос на команди и пакети помеѓу контролорот и комутаторот,

   в) протоколот *OpenFlow*, кој овозможува *OpenFlow* контролер кој може да комуницира со било кој рутер / комутатор.

2. Контролери: Контролорот може да го ажурира (ревидира, додава или избришете) проточни записи од табелата за проток во име на корисници експерименти. Статички (наспроти динамичен) контролер може да биде едноставна софтверска единица која работи на компјутер за статички (наспроти динамички) да воспоставуви патеки за пренос на IP пакети, помеѓу групата на тест компјутери за време на некој научен експеримент.
3. Влезни записи: секој запис за проток вклучува барем едноставен акција (мрежно работење) за таа проток. Повеќето OpenFlow комутатори ги поддржуваат следниве три акции:

   (а) испраќање на ова пакети на протокот на порта,

(б) инкапсулирање на овие протоци пакети и испраќање до контролорот,

(в) откажување на овие текови пакети.

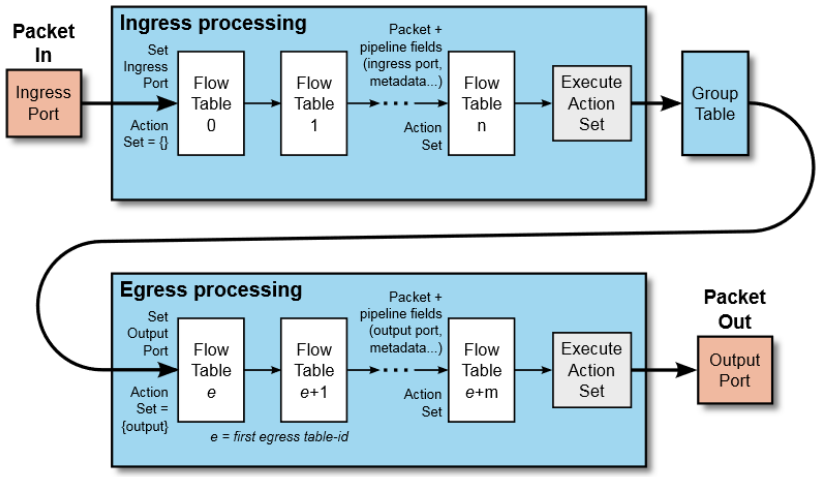
OpenFlow помина низ многу стандардни повторувања, а во моментов е на верзијата 1.5.1; сепак достапна е само верзија 1.0 за практичен софтвер и хардверски дизајн. Покрај тоа, верзија 1.6 е достапна од септември 2016 година, но достапна само за членовите на ONF (*Open Networking Foundation*).

Постојниот стандард *OpenFlow* претпоставува централизирана контрола, што всушност е следното: контролорот со од една точка може да управува со сите табели за проток во различни комутатори. Овој концепт функционира многу добро во малопродажна, кабелска локална мрежа. Кога бил *OpenFlow* предложен, бил тестиран во жичен кампус мрежа. Сепак, ако многу комутатори се распоредени во голема површина, тогаш е тешко да се користи контрола од една точка. Особено кога треба безжичниот медиум да се користи за поврзување на далечински уреди, централната контрола станува тешка, бидејќи безжичните сигнали имаат големо слабеење на долго растојание. Единствена контрола исто така има единечна точка (проблем) на испад. За да се реши тоа прашање, можеме да го користиме дистрибуирани контролери на различни локации. Секој контролер само управува со локалните комутатори. Сепак, сите контролери задржуваат високо доверливи комуникации за конзистентен поглед на глобален статус.

На Сл. 3.7 е претсавен процесот на тек на еден пакет кој ќе влезе во OpenFlow комутаторот и треба да помине низ процесот на проверка на проточните табели и групната табела. При тоа е прикажан целиот пат низ процесирачката „цевка“ со препознавање и споредување на потребните параметри за проверка/споредување, па се до самиот излез од комутаторот. Како што може да се забелеши, почнувајќи од нултата проточна табела, пакетот патува низ два процеси, влезно и излезно процесирање, кои ги дели процесирањето во групната табела.

Понатаму, да видиме што има позади *OpenFlow* стандардите, т.е. освен овој познат и најкористен протокол за SDN, постојат и други имплементации. На пример, IEEE P1520 стандардите имаат дефинирано програмибилни мрежни интерфејси [22]. Тој воедно, може да се види како почетен модел на SDN, бидејќи исто така има мрежни програмски апстракции. Понатаму ForCES (Одделување на пренасочување и контрола на елементите) [23] е уште еден стандард дефиниран од IETF. Се состои од серија на RFC-а за покривање на различни аспекти за тоа како да управуваат елементите за контрола и пренос на податоци. Тој ги

предлага моделите за одделување на IP контрола и пренасочување на податоци, мапирање на транспортно ниво за елементи за препраќање и контрола, библиотеката со блок од логички функции за таква сепарација, итн. Сепак, ForCES нема широко распространето усвојување поради недостатокот на дефиниран јасен јазик на апстракција и правилата за контролер-комутатор комуникацијата. Забележете дека ForCES има клучна разлика од *OpenFlow*: ForCES дефинира мрежни елементи и елементи за пренасочување на податоците и нивните комуникациски спецификации.



Сл. 3.7. Тек на пакети низ *OpenFlow* комутатор, [11].

Сепак, тоа не е промена на основната мрежна архитектура. *OpenFlow* ја менува архитектурата, бидејќи тоа бара рутерите / комутаторите да имаат многу едноставно препраќање на податоците и функции за рутирачката контрола треба да се отстранат до контролерите на повисоко ниво. Затоа, *OpenFlow* не може да работи во традиционалните рутери кои не поддржуваат *OpenFlow* стандарди, додека ForCES може да работи за традиционални уреди, бидејќи тоа само додава елементи за вмрежување / препраќање.

Понатаму, *SoftRouter* [24] јасно ја дефинира процедурата за динамичко врзување помеѓу мрежните елементи, лоцирани во контролната рамнина (базирана на софтвер) и податочната рамнина. Во овој стандард, мрежата може да се опише во две различни погледи, односно, физички поглед и рутирање поглед. Во физичкиот поглед, мрежата е направена од јазли во кои се поврзуваат медиумските, контролни и податочми врски. Јазлите можат да бидат елемент за препраќање или контролен елемент. Самиот елемент за препраќање е

чест рутер без локална софистицирана контролна логика. Контролниот елемент се користи за контрола на елементот за препраќање. Приказот за рутирање на мрежа ја рефлектира мрежата топологија базирана на концептот на мрежен елемент (NE, *Network Element*).

NE е логично групирање на мрежни интерфејси/порти и соодветните елементи што ги контролираат тие пристаништа. *SoftRouter* вклучува неколку протоколи: протокол за откривање (за да се воспостави обврзувачки помеѓу елементот за препраќање и контрола), FE / CE протокол за контрола, и CE / CE протокол.

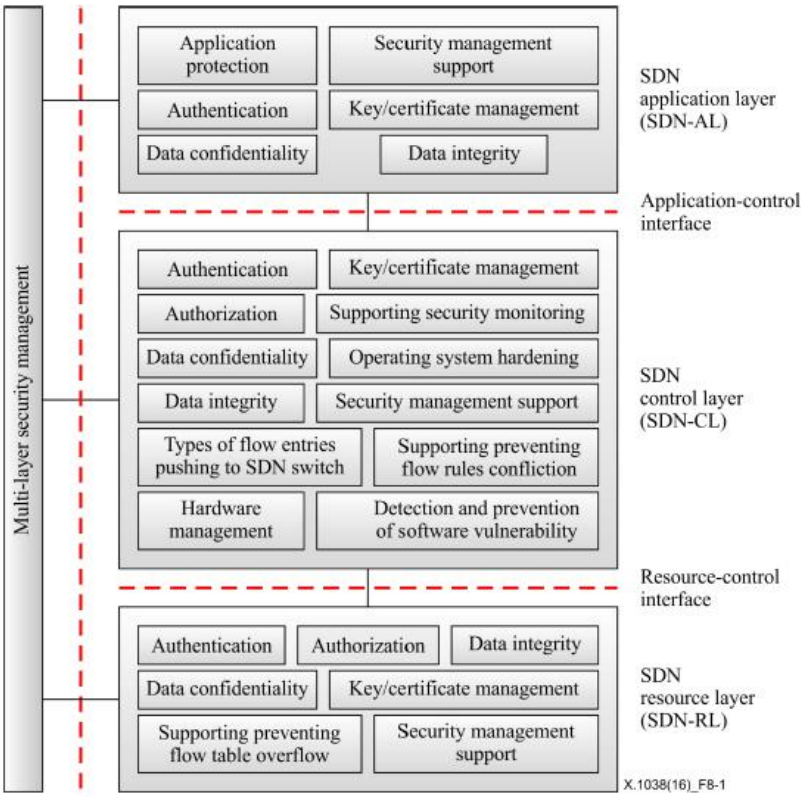
3.1.5 SDN сигурност

Како што беше претходно кажано, самото софтверско дефинираното мрежно поврзување (SDN) им овозможува на администраторите многу побрзо да ги конфигурираат мрежните ресурси и да го прилагодуваат протокот на сообраќајот низ целата мрежа за динамички да ги задоволат променливите потреби. Сепак, еден од најважните предизвици е безбедноста на SDN. Без сигурноста и безбедноста, било какви предности и QoS задоволувања се невозможни.

Така, SDN апликациите може да се гледаат како "SDN мрежни мозоци", бидејќи тие имплементираат поголем дел од мрежните функционалности. Токму затоа, нападите врз SDN апликациите, ако не престанат доволно рано или бидат спречени, може да влијаат на SDN контролниот слој многукратно и фатално.

Една од главните сигурносни и безбедносни референци за SDN несомено не води до ITU препораките т.е. Rec. ITU-T X.1038 (10/2016) [25]. На Слика 8 е претсавена таа сигурносна архитектура за SDN мрежите, при што безбедносната референтна архитектура во целост е опишана во Препорака [25] и се базира на архитектурата на високо ниво на SDN обезбедена во ITU-T Y.3300 [10].

Општо земено, безбедносните закани за SDN се заеднички и исти и за другите мрежи со традиционално вмрежување, но профилот на заканите (вклучувајќи ја и нивната веројатност и влијание, а со тоа и нивното севкупно ниво на ризик) се менува со новата SDN архитектура.



Сл. 3.8. Референтна сигурносна SDN архитектура, [25].

Со централизиран SDN контролер, влијанието на нападот од DoS (DoS)/дистрибуираниот denial of service (DDoS) може да биде повисок од оној насочен против еден рутер. Некои нови функционални ентитети (на пример, SDN контролер), протоколи (на пример, *OpenFlow*) и интерфејси (на пример, интерфејс за контрола на апликации, интерфејс за контрола на ресурси) во согласност со рамките на SDN (ITU-T Y.3300 [10]) безбедносни закани. Сите овие безбедносни закани мора да се разберат и да се решат. Препораката од ITU [25] ги опишува случаите на употреба за да ги детализира новите безбедносни закани при воведување на SDN. Освен тоа ги идентификува безбедносните закани за SDN апликацискиот слој, SDN контролниот слој, слојот за ресурси на SDN, интерфејсот за контрола на апликациите и интерфејсот за контрола на ресурсите во согласност со рамките на SDN (ITU-T Y.3300 [10]).

Оваа Препорака [25], исто така, ги дефинира безбедносните барања од анализата за заканите за безбедноста и ги проучува можните безбедносни контрамерки против нови безбедносни закани. Согласно тоа, можеме да кажеме дека безбедносната референтна архитектура за SDN е дизајнирана врз основа на идентификуваните закани за

безбедноста, и се специфицирани безбедносните барања и безбедносните контрамерки согласно тоа. Таа предложена безбедносна референтна архитектура [25] може да го води развивачот да дизајнира безбедносна функционална архитектура и да ги имплементира безбедносните функции при развивање на SDN контролер.

На кратко тука ги презентираме генералните безбедносни закани за SDN апликацискиот слој:

- *Spoofing*: напаѓачот маскиран како SDN контролер за да го добие договорот за ниво на услуга (SLA) или податоците на корисниците (на пример, идентификација на корисникот, ингеренциите) или услужна логика и да го користи за иден напад.
- Отфрлање: Корисникот или администраторот, наметнувајќи политика на злонамерен мрежа (на пример, копирање и пренасочување на специфични сообраќајни текови на злонамерен сервер), може да тврдат дека тој / таа не направи таква мрежна политика.
- Обелоденување на информации: Можно е напаѓачите да добијат ингеренции на корисникот и потоа да се маскираат како легитимен корисник за да инјектираат фалсификувани протоци во мрежа преку SDN апликација.
- Безбедносни слабости на апликациите: ранливоста на SDN апликации како што се недостатоци на кодот и несигурен код може да биде експлоатирана од страна на напаѓачот за пристап до ресурсите (на пример, SLA, податоци на корисниците, услужна логика) што ги поседува SDN апликациите за да направат натамошни напади, злоупотребувајќи SDN мрежни ресурси или реконфигурирање на целата SDN мрежа. Злонамерните апликации или недоверливите апликации од трети лица може да маскираат како правни SDN апликации за пристап до ресурсите на апликацијата.

Во Табела 1 од Препораката [25] се мапирани сигурносните побарувања со сигурносните закани, при што се дадени и насоки како да се разрешат истите.

Понатаму, генералните безбедносни закани за SDN контролниот слој се:

- Проверка на правилата на протокот: Пример за конфликт на правила за проток е опишан во употреба случај 1 од Анекс А [25]. Тој објаснува како малициозни текови би можеле да ја заобиколат детекцијата на безбедност, што е во конфликт со претходно

конфигурираната безбедносна политика и негативно ќе влијае на SDN контролорот.

- Вметнување на правилото за лажен проток: напаѓачот може да киднапира апликација SDN и да испрати некои правилни правила за протекување за да ги прислушуваат податоците. Пример е елаборирана во употреба случај 2 од Анекс А [25].
- *Spoofing*: Напаѓачот може да се претстави како администратор или SDN апликација за да ги отстрани или модифицира чувствителните податоци (на пример, податоци за конфигурација, кориснички податоци) од SDN контролерот или да добијат информации за мрежната топологија и рутирање информации или дури и да имаат целосна контрола на SDN контролорот. Со измама на адресата на SDN контролер, напаѓачот може да ја преземе контролата на целата мрежа со создавање лажен SDN контролер. Покрај тоа, напаѓачот може да создаде лажен SDN комутатор за да изврши извидување на мрежа со набљудување како контролерот реагира на различни пакети кои се генерирани од лажен SDN комутатор.
- DoS напад: Кога SDN-комутаторот наидува на сообраќај за кој нема правило за проток, тој се советува со SDN контролерот за одлука и правилото за проток за иден сообраќај од ист тип. Затоа, можно е напаѓачот да создаде измамен сообраќај за да направи DoS напади на SDN контролерот за да предизвика неуспех. Преклопен SDN комутатор, исто така, може да создаде DoS напади на SDN контролерот со постојан сообраќај за да го блокира.
- Одлагање во блокирање / ублажување на напади: Општо земено, мрежните политики се претвораат во влезни протоци за периодично да се испраќаат до SDN комутаторите во серии со цел да се подобрат перформансите на системот. Во моментов, SDN контролерот не поддржува блокирање / ублажување на нападите во реално време и SDN контролерот автоматски не идентификува кои безбедносни политики треба да работаат без одлагање. Затоа, безбедносните напади ќе траат подолго и ќе бидат потешки.
- Отфрлање: Администраторот или апликацијата за SDN, со внесување на правилата за злонамерен проток во табелата за проток за да направат внатрешни напади, може да тврдат дека тој / таа не ги внесол/ла правилата за злонамерен проток во табелата за проток.

- Обелоденување на информации: Можно е напаѓачите да добијат чувствителни информации за системот (на пример, податоци за конфигурација, кориснички ингеренции) за иден напад.
- Ранливост во оперативниот систем: SDN контролерите работаат на некоја форма на оперативни системи (ОС). Ако SDN контролерот работи на оперативен систем за општа намена, тогаш слабостите на тој оперативен систем стануваат пропусти за SDN контролерот. Напаѓачот може да ги искористи слабостите на оперативниот систем, како што се стандардните лозинки, сметки со задни врати, отворени врати (на пример, отворени порти, услуги и протоколи), па дури и без безбедносни подесувања конфигурирани да ги уништат или заменат компонентите на оперативниот систем или комплетно целиот Оперативен систем, што несомнено - сериозно ќе влијае на SDN контролерот.
- Ранливост во софтверот: SDN контролери функционираат како софтверска платформа. Ранливоста на општиот софтвер станува ранливост за SDN контролорот. А софтверската ранливост е недостаток, дефект во изградбата на софтверот, слабост или дури и грешка, која може да ја експлоатираат напаѓачите за да го сменат нормалното однесување на SDN мрежата или да ја реконфигурираат целата мрежа за понатамошни напади.
- Неуспех на хардверот: Хардверски неуспех, ништо ново со информатичката и комуникациската технологија, претставува закана за безбедноста која го претставува генеричкиот неуспех на хардверот во SDN мрежните елементи (вклучувајќи контролер и прекинувач). Хардверските неуспеси ќе ја загрозат мрежната безбедност или ќе ја снема мрежата на SDN.

Во Табела 2 од Препораката [25] се мапирани сигурносните побарувања со сигурносните закани кои се споменети за контролниот слој за SDN, при што се дадени и насоки како да се разрешат истите.

Понатаму, генералните безбедносни закани за SDN ресурсниот слој се:

- *Spoofing*: Напаѓачот може да се претстави како администратор или SDN контролер за да ги отстрани или модифицира чувствителните податоци (на пр., Податоци за конфигурација, табела за проток) од SDN комутаторот или да добие чувствителни информации, како што се влезни записи во табелата за проток.
- Прислушување: напаѓачот може да го прислушува тековите помеѓу SDN комутаторите за да види какви текови се користат, кој сообраќај е дозволен преку мрежата и кои податоци се пренесуваат.

- Обелоденување на информации: Можно е напаѓачите да добијат чувствителни информации за системот (на пример, проток, податоци за конфигурација) за иден напад.
- Прелевање на проток: Типичните SDN комутаторите имаат прилично ограничени капацитети за проток. Тесното грло на капацитетот на табелата води кон претекување на потенцијалните протоци. Затоа, можно е напаѓачот да ги пребрише легитимните правила за проток како влезни записи на табелата за проток, или да направи DoS и напади со поплави, или дури и да направи напади со инференции [26].
- Отфрлање: Администраторот или контролерот на SDN може да направи неточна конфигурација и подоцна да тврдат дека тој / таа не направил такви напади.

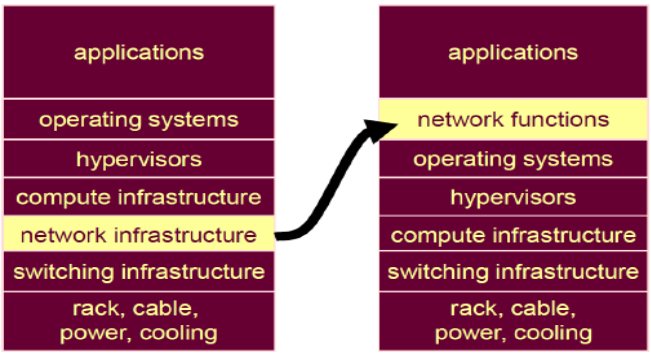
Во Табела 3 од Препораката [25] се мапирани сигурносните побарувања со сигурносните закани кои се споменети за ресурсниот слој за SDN, при што се дадени и насоки како да се разрешат истите.

3.2 NFV

Network Function Virtualization (NFV) ги имплементира мрежните функции во софтверот - кои се водат денес на сопственички хардверски стандардни сервери и ИКТ виртуелизацијата.

Како нова технологија, NFV носи неколку предизвици за мрежните оператори, како што се гаранција за ефикасноста на мрежата за виртуелни уреди, нивната динамична инсталација и миграција, како и нивното ефикасно поставување.

Додека SDN типично е наменета за управување и автоматизирачки задачи за физичките уреди, NFV е наменета да обезбедни нови вмрежени уреди. Во тој случај, SDN може да се искористи за управување на нови виртуелни уреди, покрај тоа што е задолжена за физичките уреди. На слика 3.9 е претсаена кратката споредба на SDN и каде се поместува концептот со NFV. Секако во понатамошите подглави ќе се зборуваат повеќе детали за споредбата помеѓу NFV и SDN.



Сл. 3.9. Споредба на SDN концепт (лево) по OSI нивоата и NFV концепот (десно).

Така, виртуелизацијата на мрежните функции (NFV) може да ја обезбеди инфраструктурата на која SDN може да работи. NFV е комплементарна технологија на SDN која дозволува да се изгради виртуелно-базирана од крај до крај мрежна инфраструктура и да се овозможи консолидација на многу хетерогени мрежни уреди на индустриски стандардни сервери.

Впрочем, NFV ја трансформира како мрежните оператори ја проектираат својата инфраструктура со проширување на целосно распространетата технологија за виртуелизација, за да ја оддели софтверската инстанца од хардверската платформа и со раздвојување на функционалноста од локацијата за побрзо обезбедување мрежни услуги.

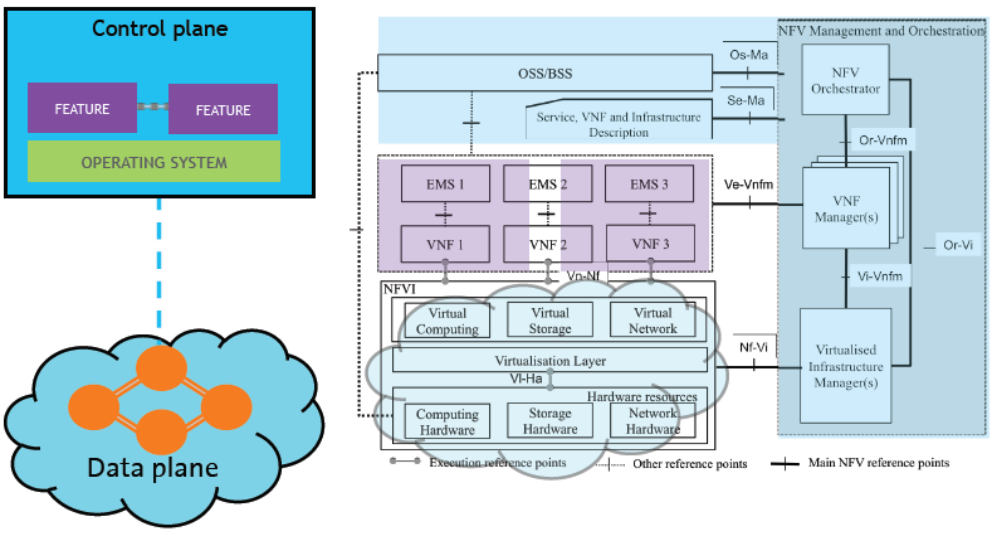
Клучна карактеристика на NFV е што мрежните функции на уредот за мрежно работење се имплементираат во софтверски пакет(и) и виртуелни машини се користат за работење со вакви пакети. Затоа NFV воведува флексибилност во распоредувањето на мрежата, бидејќи воведувањето/тестирањето на новите мрежни функционалности станува полесно: потребна е само инсталација или надградба на софтверски пакети, без да има потреба од надградба на хардверот до мрежните ентитети кои очигледно воведуваат поголеми доцнења. Како последица на тоа, NFV го намалува времето на промовирање на нови мрежни функционалности, со што се заштедува пари. Покрај тоа, NFV им овозможува на мрежните оператори да градат и оперираат на мрежа со намалени трошоци за опрема, како што генеричен хардвер може да се користи и правилно да се подесува преку софтверот според потребата на операторот.

3.2.1 Архитектура на NFV

Со виртуелизација на мрежните функции, имаме нов начин да размислуваме за крајните до крај мрежни инфраструктури. Слика 3.10 ги прикажува трите главни градбени блока на NFV архитектурната рамка

на високо ниво според Европскиот институт за телекомуникациски стандарди (ETSI). Во овој поедноставен поглед, NFV архитектурата се состои од три градбени блокови:

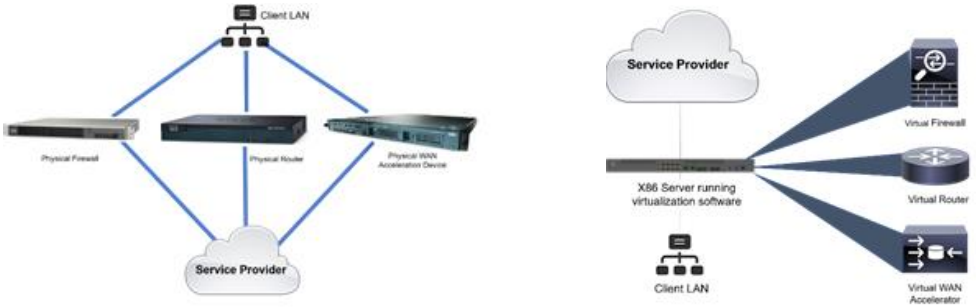
- NFV Infrastructure (NFVI): Овој блок се состои од физички ресурси (уреди, компјутери, складишта, мрежа) и нивна виртуелизација (т.е. виртуелни уреди, виртуелно компјутерирање, виртуелно складирање, мрежа).
- Виртуелизирани мрежни функции (VNF-и): Овој блок опфаќа реална имплементација на мрежни функции што се извршуваат преку NFV инфраструктурата.
- NFV менаџмент и оркестрација: Овој блок го покрива управувањето со физички и софтверски ресурси на мрежата и е одговорен за сите задачи за управување со специфичните виртуелизации.



Сл. 3.10. NFV архитектура [27].

Одделувањето на мрежните функции од посветени хардверски уреди има три главни придобивки (види Сл. 3.11). Прво, распоредувањето и работењето на мрежата станува флексибилно. Мрежните оператори можат брзо да направат промени во мрежата за да се справат со променливите барања. Тие исто така можат побрзо да ги распоредат мрежните услуги, намалувајќи го времето на пазарот. Второ, распоредувањето и работењето се помалку скапи, т.е. покажуваат помала цена на чинење. Мрежните администратори можат да користат индустриски сервери со висок волумен, наместо да купуваат хардвер кој

е со одредена специфична наменет, а тоа го прави распоредувањето на услугата и мрежата операција поефтина. Трето, виртуелизација им овозможува на давателите на услуги да излезат надвор од нивните географски пазари и да ја испорачаат услугата насекаде во светот. Виртуелизација, исто така, им овозможува на софтверот да се премести на различни локации внатре и надвор од мрежата на давателот на услуги, без потреба од инсталација на хардвер.



Сл. 3.11. Традиционален WAN наспроти NFV.

Воглавно, архитектурата на NFV, ги има следните карактеристики:

- Виртуелна инфраструктура: виртуелни машини работат на генерични хардверсервери со голем волумен, опремени со уреди за складирање и поврзани со мрежни свитчеви.
- Одвојување на софтверот: генеричен хардвер се користи од софтверот кој ги дефинира мрежните функционалности за мрежните уреди, односно хардверот не е дизајниран за специфични задачи.
- Автоматска оркестрација: оркестрацијата ја автоматизира инсталацијата и управувањето со виртуализираните мрежни функции на генеричен хардвер.

3.2.2 NFV и LINP мрежите

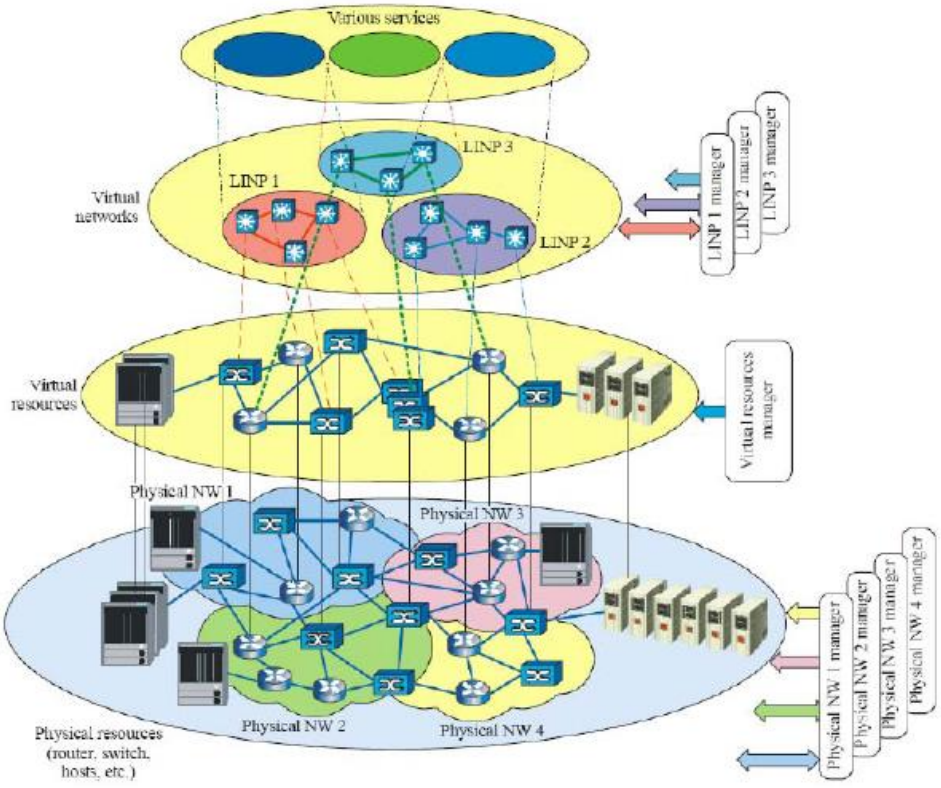
Виртуелизација на мрежата е метод кој им овозможува на повеќе виртуелни мрежи, наречени логички изолирани мрежни партиции (на англ.: *Logically Isolated Network Partitions* или LINPs), да коегзистираат во една физичка мрежа. Со цел да се обезбедат LINP-и, физичките ресурси се поделени и апстрахирани како виртуелни ресурси, а виртуелните ресурси се меѓусебно поврзани за да создадат LINP. Овие виртуелни ресурси можат да се создадат на физички ресурси, како што се рутерите, комутаторите и хостовите. Како такви, виртуелните ресурси

или се наменети за секој LINP, или пак повеќе виртуелни ресурси се агрегирани во еден виртуелен ресурс.

LINP се изолирани еден од друг, а кога се комбинираат со програмибилност во виртуелни ресурси, корисниците на LINP-ови можат да ги програмираат виртуелните ресурси на слојот за виртуелизација. Корисниците на LINP не се ограничени само на корисници на услугите или апликациите, но може да вклучуваат даватели на услуги. На пример, давателот на услуги може да закупува LINP и може да обезбеди нови услуги или технологии како услуги со пресметки во облак. Давателите на услуги можат да ги реализираат новите услуги како да поседуваат одредена физичка мрежа. Со цел да се олесни распоредувањето на виртуелизација на мрежата, неопходно е да се обезбедат процедури за контрола и управување, како што се креирање, следење и мерење на статусот на LINP мрежите.

Сл. 3.12 ја претставува концептуалната архитектура на мрежната виртуализација, која се состои од LINP над физичките ресурси кои поддржуваат мрежна виртуелизација т.е. NFV. Еден физички ресурс може да се сподели помеѓу повеќе виртуелни ресурси и секој LINP се состои од повеќе виртуелни ресурси. Секој LINP е управуван од индивидуален менаџер на LINP. На сл. 3.12, физичките ресурси во физичката мрежа(и) се виртуелизирани и може да формираат базен на виртуелни ресурси. Овие виртуелни ресурси се управувани од менаџерот за виртуелни ресурси (на англ. *Virtual Resources Manager*, т.е. VRM). VRM комуницира со физичкиот мрежен менаџер (на англ. Physical network manager, т.е. PNM) и врши контрола и управување со виртуелни ресурси. Откако LINP е конструиран со користење на виртуелни ресурси, LINP менаџерот се доделува на LINP. Менаџерот LINP врши функција за управување.

Воедно, сл. 3.12 го претставува концептот на LINP, кој се состои од повеќекратни коегзистирачки LINP-ови преку мрежните ресурси кои поддржуваат виртуелизација на мрежата. Секој LINP е обезбедена врз основа на кориснички барања. Барањата се доставуваат до VRM што ја координира распределбата на LINP, така што соодветно LINP се (или е) обезбедени за корисниците. VRM се справува со барањата врз основа на неговите административна политика. Секој LINP е контролиран и управуван од LINP менаџер.



Сл. 3.12. Концептуална архитектура на мрежна виртуализација [28]

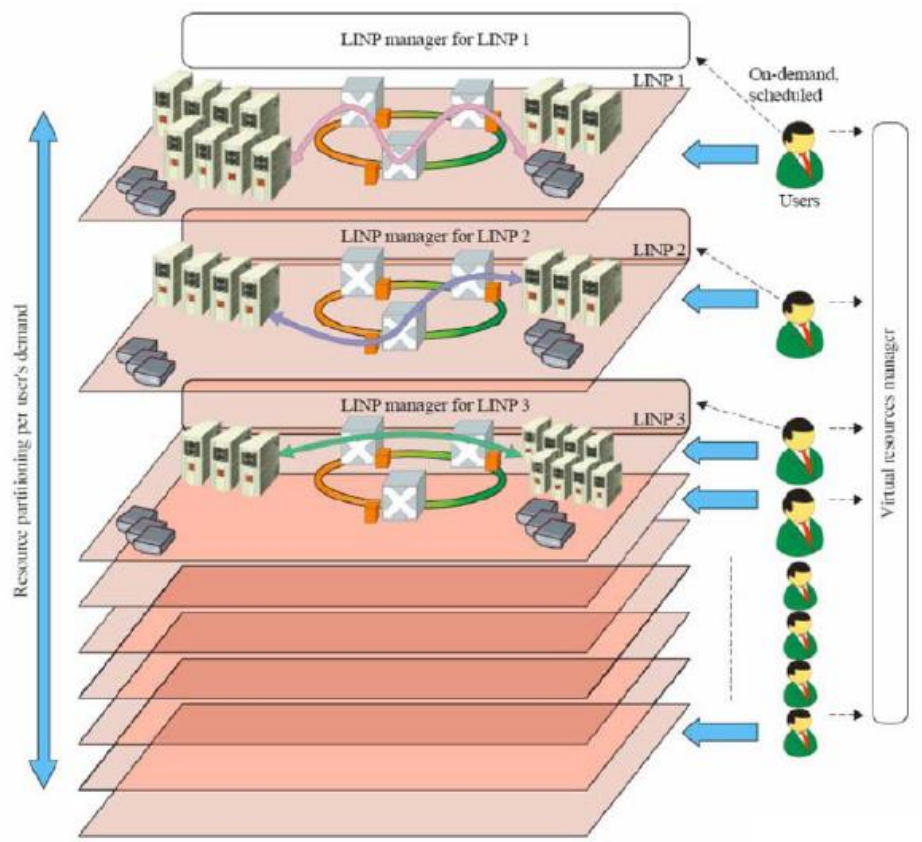
Понатаму, самиот VRM што ги контролира сите виртуелни ресурси создава LINP менаџер и доделува соодветни органи за контрола на секој LINP. LINP-овите генерирани со користење на мрежна виртуелизација има различни карактеристики, како што се:

- Партиционирање - Секоја LINP се состои од множество на виртуелни ресурси кои се независно управувани делови на физички ресурси. Повеќе LINP-ови може да постојат на една физичка мрежа.
- Изолација - Виртуелните ресурси за формирање на LINP се изолирани од тие за друг, така што LINP не смеат да се мешаат меѓусебно според перформансите, безбедноста и именскиот простор и дека било кој единствен LINP не може да предизвика нарушување на други LINP-ови или физички мрежи.
- Апстракција - Дадениот виртуелен ресурс не треба директно да одговара на неговиот физички ресурс. Деталните информации за физичкиот ресурс можат да бидат апстрахирани така што другите

системи, апликации или корисници пристапуваат до можностите на виртуелниот ресурс со користење на апстрахирани интерфејси.

- Флексибилност или еластичност - Виртуелни ресурси за изградба на LINP се флексибилно распределени, регенерирани и ослободени на побарувачката, со цел да се максимизира сместувањето на повеќе LINP на физички ресурси, за да се оптимизира користењето на физичките ресурси и временски и просторно, а исто така и за овозможување моментална и брза употреба, како и непрекината употреба на физичките ресурси.
- Програмабилност - Виртуелни ресурси за изградба на LINP може да бидат програмирани за развој, распоредување и експериментирање со нови протоколи за комуникација за иновативно ширење на податоци и за овозможување ефикасна обработка на податоци во рамките на LINP.
- Автентикација, овластување и наплата - употреба на виртуелен ресурсите за создавање на LINP мора да бидат автентицирани и овластени дека може да постигне безбедни и сигурни операции на LINP-овите, кои ја спречуваат злоупотребата на виртуелни ресурси и малициозни напади врз нив. Неопходно е да се сметаат за доделените виртуелни ресурси во физички мрежи, така што интегритетот на виртуелните ресурси може да се испита и следи, како и да има можност за оптимална употреба на виртуелните ресурси.

Виртуелизацијата на мрежата може да го подобри искористувањето на физичките ресурси, дозволувајќи му на повеќе виртуелни ресурси да коегзистираат во еден физички ресурс (Сл. 3.13). Исто така, апстракцијата и можноста за програмирање, овозможуваат стандардни интерфејси за управување и модифицирање на LINP-ови и помага да се поддржат непречена модификација и миграција на мрежата. Тоа е со цел да се обезбедат сервиси чии функции се дизајниран да биде соодветен за потребите на апликациите и корисниците.



Сл. 3.13. LINP концепт обезбеден преку мрежната виртуализација [29].

3.2.3 NFV предности

Постојат низа на предности при користење на NFV, па дел од нив се разгледани овде. Така, подолу се дадени дел од проблемите што постојат во тековните телекомуникациски мрежи, а кои можат да се ублажат со користење мрежна виртуелизација и NFV:

- Соживот на повеќе мрежи - Конвенционални технологии, како што се виртуелна приватна мрежа (VPN) и виртуелна локална мрежа (VLAN), обично се користат за обезбедување на изолирани мрежи преку споделена физичка активност мрежи. Виртуелизацијата на една мрежа може да обезбеди LINP со интерконекција на виртуелни ресурси, но меѓусебните врски може да се реализираат од различни механизми кои не се ограничени на конвенционалните механизми според корисникот и услуги барања. Исто така, мрежната виртуелизација може да обезбеди сигурно изолација меѓу LINP од различни аспекти, вклучувајќи безбедност, подобрени перформанси или

управување, и поддршка разновидност на примена, услуга, подобар QoS, мрежна контрола, управување и архитектури.

- Поедноставен пристап до ресурси - Мрежите обично се состојат од повеќекратни хетерогени физички ресурси, како што се рутери и прекинувачи, но хетерогеноста предизвикува потешкотии во пристапот и управувањето со мрежите. Мрежните оператори со цел да управуваат со цели мрежи, треба да управуваат мрежни ресурси од повеќе видови опрема, што може да ги имаат различните типови на пристапни интерфејси. Мрежната виртуелизација со NFV им овозможува апстракција на карактеристиките на физичките ресурси, така што другите системи, апликациите или корисниците можат да користат пристап до можностите на ресурсите апстрахирани интерфејси. Овие интерфејси може да гарантираат компатибилност за пристап до виртуелни ресурси и да обезбеди ефикасна контрола на виртуелните ресурси.

- Флексибилност во обезбедувањето - Флексибилноста се однесува на способноста за изградба на систем, и проширување на тоа решение, колку што е потребно, со цел да се адаптира внатрешна или надворешна промени. Во наследните мрежи, тешко е брзо да се обезбедат мрежи соодветно за барањата на различни услуги, бидејќи обезбедувањето и сигурноста на мрежите бара вистинско распоредување на физичките ресурси. Мрежната виртуелизација овозможува повторна употреба на таквите ресурси, со што се постигнува приспособливост и ефикасност во користењето на мрежата ресурси. Воедно, мрежната виртуелизација овозможува додавање или агрегирање на дополнителни логички ресурси на виртуелен ресурс, со цел да се зголеми способност по пониска цена, отколку со додавање на физички ресурси.

- Развиваемост - Ако мрежните провајдери сакаат да распоредат нова мрежна технологии и услуги или да мигрираат кон понова мрежна архитектура, тие треба да се изгради посебна тест-рамка, така што однесувањето на новиот технологии и услуги не влијае на тековните услуги. Мрежната виртуелизација може да им овозможи на мрежата лесно да се изгради логично одвоени тест-рамки, со доделување на сигурно изолирани LINP-и за експериментални цел. NFV и сеопшта, мрежна виртуелизација, која им овозможува на мрежата на услуги да се интегрираат поддршката за наследство со доделување на постоечките мрежи

на LINP. Самите LINP-ови ќе се погрижи постојните услуги и технологии да останат променети

Покрај горенаведените предности, постојат и безброј придобивки од мрежната виртуализација користејќи NFV. Дел од генералните придобивки од NFV се:

- Намалени трошоци за опрема преку консолидација на опрема за индустриски стандардни сервери со голем обем и процесорска моќ, што ги користат економиите на обемот од IT индустријата.
- Намалени оперативни трошоци: намалена моќност, намален простор, подобрен мрежен мониторинг.
- Софтверски ориентирани иновации (вклучувајќи го и софтверот со отворен код) за брзо прототипирање и тестирање на нови услуги и генерирање на нови извори на приходи.
- IT-ориентирани вештини и талент (лесно достапен во глобалниот свет, флексибилен и скалабилен).
- Флексибилност за лесно, брзо, динамично обезбедување и инстанцирање на нови услуги на различни локации (т.е. нема потреба од инсталација на нова опрема).
- Намалено време на пазарот со минимизирање на типичниот циклус на иновативни мрежни оператори. Повеќе сервисна диференцијација и прилагодување.
- Подобрена оперативна ефикасност со искористување на повисоката униформност на физичката мрежна платформа и нејзината хомогеност на други платформи за поддршка.
- Динамичко обезбедување на услуги, т.е. мрежните оператори можат динамично да ги намалат динамичните перформанси на NFV и да ја развијат потребната основа со добра контрола на грануларноста врз основа на моменталните мрежни услови.

3.2.4 Споредба на SDN и NFV

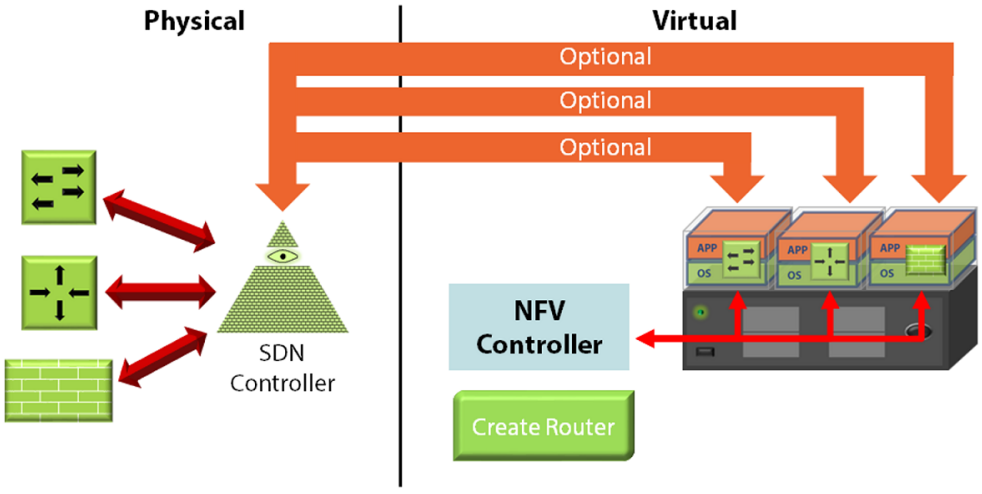
Иако имаат многу сличности SDN и NFV, кои заедно учевствуваат во виртуализацијата на мрежите, сепак има доста и разлики и секако не треба да се помешаат и идентификуваат како еднакви.

Како една од најсуштинските сличности помеѓу SDN и NFV е токму фактот што користат мрежна апстракција и виртуализација.

Во насока на разјаснување на врската и споредбата на SDN со NFV е дадена Слика 2.6. Имено, од една страна NFV распоредува виртуелна мрежна опрема и уреди, а од друга SDN традиционално управува со физичка мрежна опрема и софтвер. Секако, SDN може да се однесува на управување со физичка и виртуелна опрема и околина. На некој начин јасно е дека NFV со функционалностите е едно ниво над SDN интелигенцијата и функционалностите, затоа што ја прави виртуализацијата на едно погорно ниво.

Можеби најдобра илустрација на споредба помеѓу NFV и SDN е дадена на Сл. 3.14, каде се гледаат доста подобро распределен концепт на паралелизам во виртуализацијата и на повисока виртуелна контрола и виртуелно омрежување кај NFV. Воедно, за подобра споредба е дадено местото и улогата на контролерите и во NFV и во SDN.

Сл.3.14. SDN наспроти NFV функионалностите и обезбедувањето.



Сл. 3.15. Илустрација на SDN наспроти NFV.

Така, од една страна, SDN се обидува да ги одвои мрежните контролни функции од мрежните функции за рутирање, додека NFV се обидува да го апстрахира мрежното ниво за рутирање и други мрежни функции од хардверот на кој работи. Така, и двете зависат во голема мера од виртуализацијата, за да се овозможи мрежен дизајн и инфраструктура, со цел да се апстрахираат во софтверот, а потоа да се имплементираат од основниот софтвер на хардверските платформи и уреди.

Кога SDN извршува на NFV инфраструктурата, SDN пренасочува пакети со податоци од еден мрежен уред на друг. Во исто време, функциите за мрежно контролирање на SDN за рутирање, дефинирање на полиси и апликации - работаат во виртуелна машина, некаде во мрежата. Така, NFV обезбедува основни мрежни функции, додека SDN ги контролира и ги оркестрира за специфични намени. SDN понатаму дозволува конфигурацијата и однесувањето да бидат програмски дефинирани и променети.

SDN и NFV се разликуваат во тоа како ги одвојуваат функциите и апстрактните ресурси. SDN апстрахира физички мрежни ресурси - комутатори, рутери и така натаму - и го движи донесувањето на одлуки од виртуелната мрежна до контролната рамнина. Во овој пристап, контролната рамнина решава каде да испрати сообраќај, додека хардверот продолжува да го управува и да го насочува сообраќајот. NFV има за цел да ги виртуелизира сите физички мрежни ресурси под хипервизорот, што овозможува мрежата да расте без додавање на повеќе уреди.

Може да кажеме дека SDN и NFV се поврзани, но не се заменливи и не се комплетно комплементарни. Тие се различни, но многу комплементарни технологии кои можат да се користат, или сами засебно или во комбинација. NFV може да се имплементира без да се бара SDN, и обратно - мрежните оператори можат да изградат SDN без мрежна функциска виртуелизација (NFV). Меѓутоа, двата пристапи може да се комбинираат и ова комбинацијата нуди силна синергија за мрежните оператори и провајдери на Интернет услуги.

Кратко кажано - додека и SDN и NFV ги прават мрежните архитектури пофлексибилни и динамични, тие извршуваат различни улоги во дефинирањето на тие архитектури и инфраструктурата што ја поддржуваат.

3.3 Примена на SDN & NFV во паметни градови и паметни услуги - Свет на паметни сервиси

Во 2015-та година, околу 15 милиони уреди во светот беа закачени на Интернет. До 2020 година, овој број веројатно ќе се зголеми на 30 милијарди. Високо автоматизираната технологија овозможува големи протоци на податоци кои треба да бидат анализирани, обработени и комбинирани на нови начини за креирање иновативни паметни услуги. Паметните услуги претставуваат основа за формирање на нова класа на производство на корисни мрежи кои отвораат нови можности за производство, развој и дизајн на нови производи и услуги. Ова исто така значи дека тие може да се користат за креирање на нови бизнис модели. Користени во приватните домови, паметните услуги имаат потенцијал да обезбедат поголеми нивоа на безбедност, погодност и енергетска ефикасност. Паметните услуги се сервисни платформи кои ги соединуваат програмерите, операторите и корисниците. Како такви, тие се предмет на одредени законски барања, а не само на правилата за приватност на податоците.

Во таа насока се наметнуваат нужно некои прашања: Паметни, поврзани или IoT (*Internet of Things*)- базирани уреди? Која е разликата? Дали човештвото станува попаметно со паметни уреди, паметни технологии и паметни градови? Ако сè околу нас станува попаметно, тогаш зошто треба да го притиснеме копчето? Со преоптоварување на информации насекаде - дали овие нови паметни поврзани уреди ги олеснуваат нашите животи? Дали сега се чувствуваме попродуктивни и поздрави? Дали забележавте како различни производители ги продаваат овие уреди? Некои се паметни, други се поврзани, а некои се базирани на IoT. Постои одредена збунетост околу разликата меѓу нив и нивната дефиниција се преклопува премногу.

3.3.1 Паметни уреди

Паметниот уред е електронски уред, генерално поврзан со други уреди или мрежи преку различни безжични протоколи како што се: Bluetooth, NFC, LiFi, 3G, 4G, идни 5G и така натаму, кои, до одреден степен можат да работат интерактивно и самостојно. Неколку значајни типови на паметни уреди се: паметни телефони и таблети, паметни часовници, паметни звучници и слично. Терминот, паметен уред, исто така може да се однесува и на уред кој покажува некои својства на сеприсутното компјутерско работење, вклучувајќи, иако не е морално, вештачка интелигенција.

Паметните уреди можат да бидат дизајнирани така да поддржуваат различни фактори, голем број на особини кои се однесуваат на сеприсутното компјутерско работење и се користат во три системски средини: физички свет, средини со фокус на човекот и дистрибуирани компјутерски средини. Воедно, истите, обезбедуваат одредено ниво на автоматизација и може да бидат програмирани за некои специфични употреби.

Иако флексибилноста на конфигурацијата е ограничена, тие се брзи и ефикасни во извршувањето на она што се очекува. Како на пример, паметната машина за кафе - со притискање на неколку конфигурабилни копчиња може да ве разбуди со готова шоља на топло кафе. Друг пример е паметен термостат кој ја одржува температурата на вашата соба. Порано, паметните уреди не беа потребни за поврзување на мрежа и се користат за да работат самостојно, но денес повеќето паметни уреди се поврзани- заматувајќи ја линијата меѓу паметниот и поврзаниот уред. Паметна сијалица, паметен звучник, паметен термостат или паметни безбедносни камери се поврзани уреди.

3.3.2 Поврзани уреди

Далечински контролирани и управувани мрежни уреди, со еден збор се нарекуваат поврзани уреди. Така, една мобилна апликација на нашите мобилни телефони може да се користи за поврзување на уредот. Можете да го подготвите кафето со давање инструкции од вашиот мобилен телефон и откако кафето ќе биде подготвено, ќе добиете известување. Вие може да поставете арома, време за пиење и слично од вашиот мобилен телефон.

Поврзувањето може да биде преку Bluetooth, LTE, WiFI или жица. Паметните и поврзаните уреди се користат заедно [30].

3.3.3 IoT уреди и Cloud

Всушност, IoT уред претсавува: Софтверски- дефиниран производ или дигитален пар со:

- производ,
- апликација, 3
- анализа
- и Интернет (или мрежа).

IoT уредите создаваат паметни градови, памтни фабрики, паметни лаборатории, паметни училници или паметни домови. Добивате паметни брави за врати кои може да бидат отворени со мобилен телефон, трагач за патни торби за да не ја изгубите вашата чанта за лап топ на аеродромот, AWS IoT програмабилни копчиња (види Сл. 3.16).

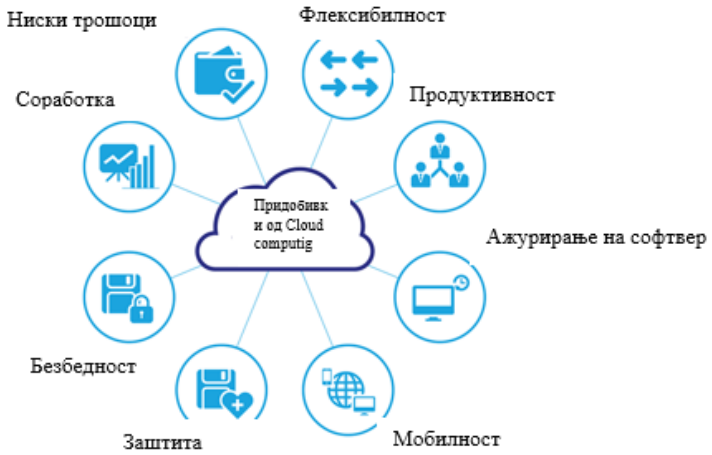
Тие создаваат поголема вредност отколку паметни или поврзани уреди - затоа што се повеќе скалабилни, надградливи, автоматски и подготвени за иднина [30].

Сл. 3.16. Илустрација на AWS IoT копче.

Можете да ја кодирате логиката на IoT копчето во cloud за да ги конфигурирате кликнувањата на копчињата со цел да броите или следите предмети, повикате или алармирате некого, да започнете или да прекинете со нешто, да нарачате услуга, па дури да добивате повратни информации. На пример, можете да кликете на копчето за да отклучите или стартувате автомобил, да ја отворите вратата од гаражата, да ја следите употребата на заеднички домашни работи, лекови или производи, или далечински да ја котролирате вашата домашна апаратура [30].

Паметните уреди извлекуваат многу лични податоци од вашиот телефон, паметна телевизија и ладилници. Друг проблем е, бидејќи овие уреди се поврзани со мрежа користејќи интернет и облак (cloud) - ова предизвикува висок ризик или протекување на приватни податоци и други слабости.

Замислете, инсталиравте некоја апликација на вашиот паметен телевизор и таа апликација може да користи некои позадински, потребни информации , а потоа, да ги користат и други домашни уреди низ *Cloud* услуги (види сл. 3.17).



Сл. 3.17. Придобивки од *Cloud* услугите.

Тоа се технологии во кои големи, скалабилни ресурси, како што се процесирачка моќност, меморија, се обезбедени како онлајн услуги. Наместо корисниците да ги имаат во сопственост и да бидат одговорни за овие ресурси, тие можат лесно да пристапат до нив, користејќи Интернет, веб пребарувач, или специјални апликации и софтвер. Примери на често користени cloud услуги вклучуваат web mail (*Gmail*), online data storage (*Dropbox*) и CRM *systems* (*Salesforce.com*). Денес, околу половина од сите фирми ги користат cloud услугите [31].

Во зависност од услугата, cloud услугите се поделени во три категории и тоа:

- SaaS (*Software-as-a-Service*): Корисниците добиваат пристап до софтверот на мрежата на обезбедувачот на cloud услугите и не мораат да ја менуваат постоечката инфаструктура, како што се хардверот, оперативните системи, меморијата и слично. Примери се *web mail* и *Google Docs*.
- PaaS (*Platform-as-a-Service*): Корисницте, најчесто програмери или развивачи на системи, добиваат пристап до различни околини за развој за да создадат нивни сопствени апликации и софтвери. Примери се MS *Azure* и *Google App Engine*.
- IaaS (*Infrastructure-as-a-Service*): Добавувачот ја обезбедува процесирачката моќност, меморијата и мрежните компоненти кои што се од големо значење за корисниците да можат да инсталираат и користат различен софтвер, вклучувајќи и оперативен систем. Примери се *Oracle, VirtualBox* и *VMware*.

Понатаму, *Cloud* услугите имаат три различни модели за дистрибуција:

- Јавен (*Public*) *cloud*: Услугата е достапна за јавноста, а е во сопственост на провајдерот.
- Приватен (*Private*) *cloud*: Инфраструктурата и вклучените услуги се користени од страна на само една фирма или организација. Тие може да бидат менаџирани интерно или преку надворешен вендор и да бидат сместени и во интерните и екстерните компјутерски центри.
- Хибриден или мешан (*Hybrid*) *cloud*: Комбинација од повеќе cloud услуги, што овозможува нивна интеграција.

Се очекува *cloud* услугите да добијат посилна позција, бидејќи паметниот град и неговите актери може да ги користат овие услуги за да ги избегнат првобитните трошоци и да осигураат поголема флексибилност. При бирањето на екстерни актери, секој град мора да

има јасно разбирање за сопственоста на податоците за градот, бидејќи можат да се појават проблеми поврзани со личната сопственост доколку неовластени лица имаат пристап до податоците на градот [31].

3.3.4 Паметна мрежа

Паметна мрежа претставува сложена електрична мрежа која сигурно и квалитетно ги поврзува и обезбедува оптимално функционирање на сите составни елементи, од генераторот, до метеоролошки систем, систем за усогласување на понудата и побарувачката, систем за мрежно поврзување, па се до административен систем кој е во директен контакт со потрошувачите на електрична енергија. Воведувањето на паметни мрежи го подобрува управувањето со енергија и поттикнува ефикасно искористување на енергијата во зградите.. Истовремено, енергетските компании ќе знаат колку енергија треба да се испорача, што резултира во намален дел од неискористената енергија. Потребен е целосен пристап во кој сите процеси ќе бидат водени од употребата на модерните ICT технологии со постигнување на максимална енергетска ефикасност и минимално загадување на околната средина. Целта е од централизиран систем за производство на електрична енергија, да се развие децентрализиран систем за производство на електрична енергија врз основа на обновливи извори на енергија.

3.3.5 Паметни мерила

Паметните мерила во редовни интервали ја регистрираат потрошената електрична енергија или гас и податоците автоматски се испраќаат преку мобилна или фиксна мрежа. Придобивките од паметните мерила се точните податоци за потрошениот ресурс, а не се сметки со проценка за потрошувачката, Бидејќи ние во секое време можеме да го отчитаме потрошувачот, поголема е можноса за оптимизација на потрошувачката на гас или енергија во текот на целата година. Врз осова на добиениот профил за потрошувачката, добавувачите можат да понудат индивидуални тарифи кои ќе доведат до ефикасна потрошувачка на енергетските ресурси со позитивни последици за животната средина и здравјето на луѓето.

3.3.6 Паметни згради (*Smart Building*)

Бидејќи зградата е основна градежна единица на градот, најпрво треба да почне да се менува од пасивен потрошувач на електрична енергија во активен учесник кои собира податоци и создава нови информации. Тие информации ќе служат за управување со снабдувањето и потрошувачката. Со примена на систем за паметна координација потрошени на ниво на зграда, на ниво на мрежа може да се предвиди

намалување на потрошувачката што е неопходно за време на врвните периоди. Паметните згради, покрај управувањето со внатрешните услови, имаат способност да комуницираат со околината и да ја прилагодуваат сопствената потрошувачка на мрежно ниво. Исто така, паметните згради имаат услови за остварување на меѓусебна комуникација, создавајќи активни локални мрежи кои, покрај трошоците, може да вклучат мали обновливи и класични извори на енергија. Со интегрирање на обновливите извори, ситуацијата станува покомплицирана, бидејќи зградата станува активен елемент на мрежата. Создавањето на паметни згради и нивното поврзување ќе создаде локална мрежа на активни потрошувачи, а меѓусебото поврзување на таквите локални мрежи ќе создаде паметна мрежа [31].

3.3.7 Сервисни сегашни и идни трендови

Неколку апликации често се споменуваат во врска со терминот „Паметен град“. Во моментот нема поделба на апликациите кои ги користат сите актери, но постојат неколку теми кои се повторуваат кога ќе се спомене паметен град. Вакви апликации се следните:

- Паметни услуги за сообраќај
- Паметни услуги за енергија
- Паметни згради
- Паметни ресурси
- Паметно водење на грижа за здравјето
- Паметна администрација

Во контекст на развојот на паметни градови, важно е да се разберат потенцијалните последици од изборот на модел на пазарот за условите на паметните градови за стимулирање на развој на услуги, ефикаснот, флексибилност, иновација. Во врска со брзиот развој на решенија за паметни градови, важноста на избраниот модел на пазар значително ќе порасне. Различни модели ќе постојат на пазарот и различни актери ќе поддржуваат различни модели. Може да се каже дека моделот на пазар е основа врз која се бира изградба на паметен град. Два различни трендови за модели на пазарот се: модел за вертикална интеграција и lock-in и конзорциум и партнерство.

Трендовите на модел на пазарот влијаат на актерите кои работат во областа на паметните градови, каде што важно е:

- Трендови на модел на пазарот се движат кон повеќе вертикални решенија наменети за заклучување на клиенти. Иако актерите на пазарот формираат конзорциуми, ова го ограничува бројот на актери и ја ограничува конкуренцијата.
- Пазарот се развива кон специјалистички ориентиран сектор за дигитална услуга, кој само по себе го движи техничкиот развој [32].

3.3.8 Паметни системи за транспорт

Голем број на автомобили и други возила кои за погон користат фосилни горива се причина за зголемено загадување на воздухот, емисии на стаклени гасови, пренатрупаност и каснење. Решението кое има за цел да ги намали гореспоменатите последици е воведувањето на т.н. паметни сисеми за транспорт. Тие се одесуваат на процесот на производство на возила, на подобрување на комуникациската мрежа меѓу возилата т.е. V2V (Vehlice to Vehlice), како и помеѓу возило и инфраструктурна мрежа т.е. V2I (*Vehicle to Infrastructure*).

Во паметните градови, кај примената на интелигентните(паметни) системи се очекува дека во реално време:

- ќе ги оптимизира сообраќајните патишта, а со тоа и протокот на сообраќајот на патот,
- ќе овозможи лесно бирање помеѓу различни типови на возила,
- има позитивно влијание врз процесот на производство на возила, каде што по потреба на паметните градови ќе се вградат и нови функции,
- зголемување на протокот на луѓе и стока во сообраќајот.

3.3.9 Енергетска ефикасност

Енергетската ефикасност е една од петте цели во стратегијата за развој на Европа 2020, воедно дел и од развојот на идните 5G мрежи, а се однесува на енергетска одржливост и климатски промени.

Се планира да се постигне:

- намалување на емисиите на стакленичките гасови за 20% во споредба со 1990 година,
- 20% енергија од обновливи извори на енергија,
- зголемување на енергетската ефикасност за 20%.

Европската комисија иницирала поттикнување на стандардизацијата, особено во паметните мрежи, што претставува ефикасен пристап за постигнување на наведените цели. Создавањето на паметен град не е нешто што може да се појави преку ноќ, и покрај добри и ненадејни инвестиции. Создавањето на паметен град е концепт кој се развива со години и се имплементира колку што му дозволува технолошкиот напредок и животната средина.

Паметните градови стануваат реалност, поради огромното истражување на технолошките овозможувачи и развојот на IoT, што овозможува многу апликации и сервиси кои се изградени околу различни видови на сензори. За да се справат со зголемената популација во градовите, има потреба од поодржливи, еколошки и економски - пријателски попаметни градови и технологии. Неодамна, бројот на паметни уреди се појави во големи размери, како што е *SleepNumber*-паметен кревет со фокус на здравјето на луѓето за време на спиењето, паметната четка за заби *Kolibree* и *Belkin* – паметната тава. Ова се неколку, со повеќе апликации и продукти кои ќе бидат прикажани во блиска иднина.

Во контекст на ова посебно прашање, дефинираме паметен град како збир на субјекти (живеење и неживеење) во урбана област која секогаш е поврзана, целосно свесна, автоматизирана, само-сигурна, адаптивна и добро информирана. Освен тоа, зголеменото присуство на широкопојасни мрежи со ултра-големи брзини, сеприсутни безжични мрежи, *cloud computing*, сензори и софтверско-дефинирана инфраструктура поврзуваат паметни/ мобилни уреди за да генерираат релевантни податоци за градовите, во огромни размери. Овој напредок ќе овозможи апликации и услуги кои ќе го подобрат квалитетот на животот на луѓето при решавањето на важни национални преоритети како што се следење во реално време, безбедност, автентичност и достапност на класифицирани информации до донесувачите на одлуки. Слично на тоа, за да се направи паметен град, потреба е цврста комуникациска инфраструктура за поврзување на паметни објекти, луѓе и сензори.

Комуникацијата во градовите вклучува повеќе пристапни мрежи кои можат да бидат јавни или приватнии. Градот може да собира податоци од паметните уреди и сензори вградени во патиштата, паметни мрежи, згради и други средства. Ги споделува тие податоци преку паметен систем за комуникација, кој обично е комбинација од жични и безжични мрежи. Потоа користи паметен софтвер за да креира вредни информации и дигитални услуги, како што се помош со здравството, безбедност и

сигурност, следење на сообраќајот во реално време и управување со животната средина.

Во ова посебно прашање на енергетска ефикасност, целта е да се здружат научниците, академците и поедиците кои работаат на посебни области на паметни градови, и во IoT, па заедно со новите технологии да ги споделуваат своите нови идеи, најнови сознанија и резултати. На еден таков начин би се постигнале многу цели околу енергетската ефикасност.

3.3.10 SDN и NFV во паметни градови и паметни нешта

За да се реализира визијата за паметен град (како и се од погоре споменатото како нова технологија и IoT концепти), иницијативата за паметни нешта (или Интернет на се) треба да биде поддржана и интегрирана од интелегентна комуникациска инфраструктура, како што е SDN и NFV. Методите на SDN и NFV овозможуваат подобра управливост, интеграција и пристапност, кои се клучни барања за да се постигнат реални придобивки за градот како целина. Едноставно кажано, идејата за паметните градови е развиена со цел да се поттикне подобра интеракција меѓу луѓето, просторот и нештата. За да тоа се случи, ќе биде потребна инвестиција во широк спектар на технологии, вклучувајќи ги сеприсутните фиксни и безжични мрежи за широкопојасен пристап (5G), IoT технологии, M2M комуникации и големи алатки за податоци. Исто така,со сегашните 6GB уреди поставени да станат 25GB во 2020 ќе има реални барања за поефикасен пренос на податоци, и што е најважно помали трошоци.

Сценариото на паметни градови поставува неколку предизвици во управувањето со ресурсите на мрежата. Околината на паметните градови се очекува да биде хетерогено сцеарио каде различни видови на уреди (паметни телефони, сензори, актуатори и сл.) коегзистираат во хетерогени употреби (пример, макро, пико, фемто-клетки) и имаат хетерогени сообраќајни шеми (на пример, комуникацијата меѓу машини бара висока доверливост и сигурност и ниска латентност за да се намали потрошувачката на енергија додека сообраќајот ориентиран кон човекот има помали побарувања во поглед на потрошувачката на енергија). Оваа внатрешна хетерогеност во околините на паметните градови бара брза реконфигурација на мрежните параметри/распоредувања во согласност со сегашната состојба на мрежата: ова јасно ја покажува неефикасноста на тековите стратегии за распоред усвоени од мрежните оператори, главно базирани на претходно конфигурирана параметризација на мрежата и ad-hoc мрежни уреди со предефинирани задачи. Во 5G системите, мрежата треба да биде конфигурирана во согласност со

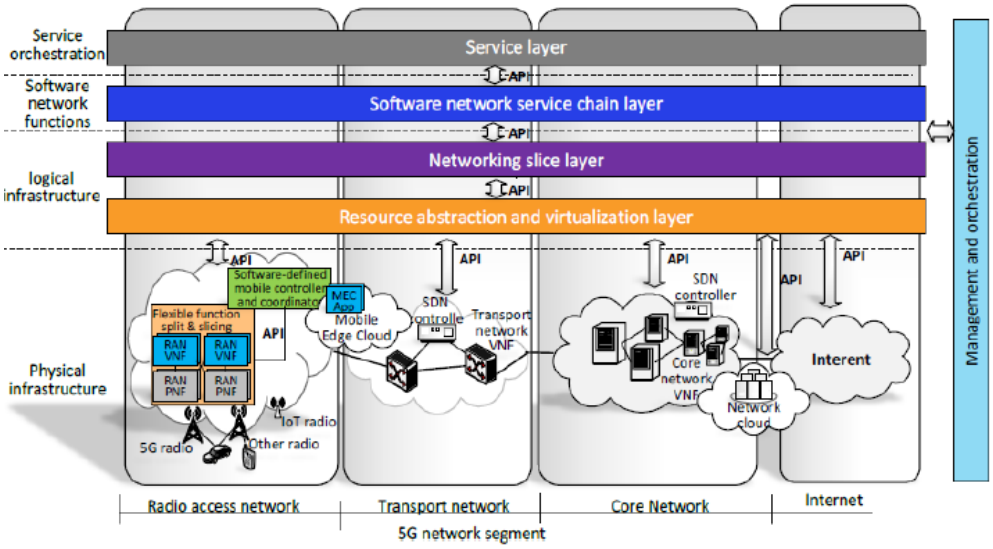
иформации за сообраќајот, нивоа на мобилност, нивоа на интерференција, QoS барања, преоптоварување на радио/јадрените сегменти и така натаму. Ваквите информации очигледно се менуваат со време, па оттука се диктира за нови решенија овозможувајќи ниска латентност во реконфигурацијата на мрежата. Горенаведените дискусии за софтверизација и виртуализација на мрежата се воедно корисни и за постигнување флексибилност дека паметните средини претставуваат 5G системи. Примери за подобрувањата воведени со SDN/NFV (*Software-Defined Networks/ Network Function Virtualization*) се дадени во [33] и се сумирани тука:

- динамичка конфигурација на клетка, сообраќаен баланс и управување со ресурси;
- најдобри меѓусебни врски меѓу мрежните уреди;
- најдобри врски меѓу предавателите и физичките елементи;
- активирање на соодветните примопредаватели кои ќе бидат вклучени во справување со одредена ситуација.

Сепак, за постигнување на овие цели треба да се земат во предвид неколку нешта и аспекти. Прв аспект е потребата од динамичко пренасочување на корисничкиот сообраќај кога се врши прилагодување на услугите: ова станува предизвик, бидејќи сеуште не е јасно како постоечките SDN контролери работаат во широка област (јадрото) на 5G мобилните системи. Кога се разгледува оптоварувањето значајно за паметните градови, каде огромниот и непредвидлив број на уреди се очекува истовремено да се поврзат во ограничена зона, приспособливоста станува загриженост за избегнување на преоптоварувањето на мрежата и доцнењето. Покрај тоа, кога се фокусираме на апликации каде сензорите и актуаторите треба да комуницираат под строги ограничувања во доцнењето, оптоварувањето може да вклучува неприфатливи доцнења кои можат да предизвикаат нестабилност во некои сегменти од паметниот град.

Друг интересен предизвик е натамошното редуцирање на апликациите, кои се сметаат за примарни услуги за паметни градови. Комуникациите својствени за таквите апликации се однесуваат на трансмисијата на многу ограничен сообраќај (неколку бајти) чие управување со сегашниот 3GPP стандард вклучува користење на носечки ресурси во јадрената мрежа. Натамошното редуцирање доведува до потреба од нов дизајн на протоколните интерфејси во SDN/NFV во 5G архитектурата (види Сл. 3.18) со цел да се гарантираат придобивки за евтини сензорски уреди (т.е заштеда на енергија како што е потребен

помал број на контролни бити за за секој пренос на податоци) и во радио/јадрени мрежи (т.е потребна е помала количина на податоци и контролни ресурси за податочните/контролни носители).

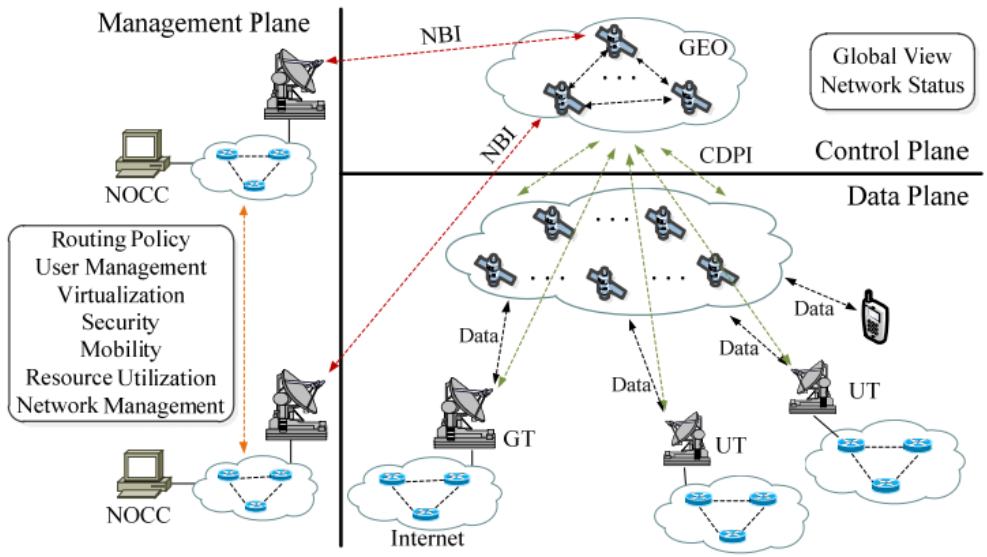


Сл. 3.18. SDN/NFV во 5G архитектурата [34].

Конечно, од голема важност е безбедноста. Во NFV мрежата, виртуелните апликации се извршуваат во податочни центри кои не може директно да бидат поседувани од мрежните оператори. Покрај тоа, воведувањето на оркестратори може да генерира дополнителни безбедносни слабости со што поголеми оптоварувања (и последнователно-повисоки одложувања) на системите и/или функционалностите за детекција на упад. Безбедносните закани исто така се должат на употребата на заедничко вмрежување и складирање, односно кога виртуелните машини ги споделуваат физичките ресурси со друга мрежни апаратури или кога компонентите базирани на софтверот се нудат на различни вендори; овие сценарија потенцијално може да создадат безбедносни дупки поради комплексноста на интеграцијата. Како последица на тоа, операторите треба да бидат сигурни дека безедноста на нивната мрежа во иднина нема да биде засегната од погоре разгледаните прашања, па потребно е да се преиспитаат безбедносните прашања при дизајнирање и имплементација на 5G со SDN и NFV системи.

3.4 Примери за примена на SDN во 5G и сателитски мрежи

Со примената на ICT решенија во процесот на производство, пренос и дистрибуција на електрична енергија, како и зголемување на енергетската ефикасност во домаќинствата, зградите, транспортот и индустријата, постои премин кон енергетски-помалку барања за услуги и производи. Воедно, значително го намалува загадувањето на околината, како и емитирањето на стаклени гасови. Со развивање на нови супервизорски стратегии, се стремат да обезбедат динамично и ефикасно управување со енергетски ресурси земајќи ја во предвид промената на природата на изворите на енергија. Неопходно е да се постигне согласност во поглед на архитектурата на ICT комуникациската мрежа која ќе ги задоволи потребите на различни услуги. Широкопојасните системи, 5G, размената на се поголемата количина на податоци меѓу произведителот, потрошувачот и самата мрежа, ја прават се посигурна т.е потребно е да се обезбеди ефикасна контрола на квалитетот на испорачаната енергија и интерактивната комуникација со крајните корисници.



Сл. 3.19. Пример на SDN сателитска мрежа [35].

Сликата 3.19 означува една постоечка архитектура на SDN сателитска мрежа, која е составена од контролна, податочна и управувачка рамнина.

- Податочната рамнина се состои од крајни рутери дистрибуирани низ светот и мултинивовска сателитска инфраструктура (GEO, MEO, LEO). Сателитите и рутерите користат flow table “match-action” протокол и се фокусираат на препраќање на пакети. Протоколот го таргетира хедерот на секој пакет како IP адреса, порта и кориснички сегмент за подршка на препраќање , мултикаст, виртуелна мрежа, менаџмент итн. Мултинивовскиот систем има засебни карактеристики. На пример, GEO сателитот подлежи на големо доцнење, но линкот е доверлив. Од друга страна, доцнењето кај LEO сателитот е многу помало.Така со избор на различни рути, мрежата може да гарантира квалитет на сервис за различни видови на сервиси.
- Контролната рамнина е управувана од GEO сателит, бидејќи тој нуди доверливост на линк, голема покриеност и бродкастинг. Доволни се 3 GEO сателите за да се покрие целата површина, па така бројот на GEO контролери се сведува на три. Оваа група на контролери е логички централизирана единица која се фокусира на толкување на правилата кои стигаат од управувачката рамнина до податочната рамнина, мониторирање на статусот на мрежата и препраќање на истиот до управувачката рамнина за да се добие апстрактен поглед на мрежата. Во споредба со традиционалните мрежи, оваа SDN архитектура го намалува бројот на терестријални станици и го поедноставува процесот за контролата на поток. Како што е прикажано на сликата 3.19, постојат три топологии за групата на контролни GEO сателите, кои се разликуваат по комплексноста и доверливоста.

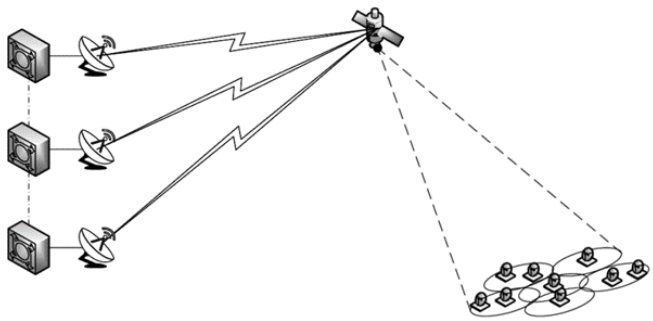
Сл. 3.19. Различни топологии.

- Управувачката рамнина извршува различни модули за различни апликации, како калкулација на рутирачки полиси, виртуелизација, сигурност, менаџмент на ресурси. Апликациите зависат од статусот на мрежата. Како за пример, при

прикулучување на нов мобилен терминал кон мрежата, управувачката рамнина треба одново да ги пресмета рутирачките полиси и да ја предадае табелата за потокот кон податочната рамнина.

3.4.1 Поставување на SDN контролери и gateway-и во SDN овозможена сателитска мрежа како дел од 5G системот

И покрај интеграцијата на терестријалните 5G и сателитски мрежи носи многу придобивки, сепак донесува и многу предизвици. Прво, како да се одреди местоположбата на терестријално-сателитската опрема за рутирање, т.е така наречени сателитски gateways треба да бидат добро објаснети. Второ, во интегрираната мрежа, обично ќе има многу достапни патеки низ кои податочниот сообраќај може да се рутира. Кога и како да се избере патека, притоа избегнувајќи оптоварување на мрежата и најефикасно искористување на ресурсите мора внимателно да се разгледа.Трето, кога квалитетот на сателитските ликови ќе се влоши под слаби метеоролошки услови, протокот на податоци треба да се трансформира од еден сателитски gateway до друг. Како да се осмисли паметен механизам за хендовер за да се гарантира квалитетот на врската е друг предизвик. Покрај тоа, 5G-сателитската интегрирана мрежа е составена од различни типови на инфраструктури, што резултира во хетерогени мрежи (HetNets) и како да се обезбеди флексибилно и програмабилно управување со овие мрежи е исто така важно прашање. Сите овие предизвици можат да бидат добро решени со усвојување на *Software-Defined Networking* (SDN). Примената на SDN во 5G-сателитската интегрирана мрежа овозможува управување со целата мрежа преку интелигентни системи за управување и оркестрација. Во SDN, логичката контролна единица е одвоена од комутаторот за централизирана контрола во податочната рамнина. SDN контролерот (сл. 3.20) има целосен преглед на мрежата, така што е во состојба да носи одлуки на глобално ниво.



Слика 3.20. Пример за сателитски контролер.

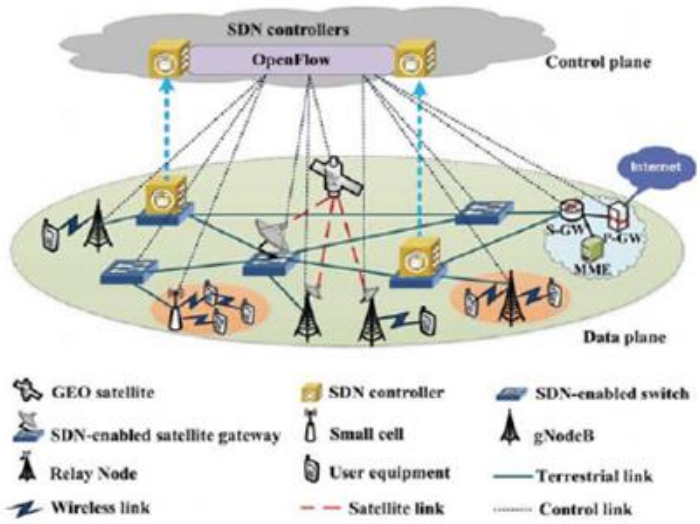
Врз основа на ограничувањата на протокот и индикаторите за мрежни перформанси, контролерот може лесно да донесе одлука како да го спроведе хендоверот меѓу сателитските *gateways*. Уредите во HetNets не треба да разбираат различни протокли, туку само да примаат инструкции од контролерот, со што се отвара патот за управување со овие уреди со голема флексибилност и програмабилност. Значи, контролерот има значајна улога во SDN-овозможената 5G сателитска интегрирана мрежа. За да се обезбеди правилно функционирање на свитчевите или сателитите, неопходно е да се одржува добра конекција меѓу овие јазли (комутатори или сателити) и контролери, инаку тие ќе примаат/праќаат застарени информации од/кон контролерите. Повеќето мрежни компоненти, како што се терестријалните јазли и ликови и сателитските врски во интегрираната мрежа, се склони кон неуспех поради многу фактори. Затоа, со цел да се подобри сигурноста на интегрираната мрежа, од фундаментално значење е да се одреди поставувањето на контролерите и сателитските gateway-и. Генерално, при одредувањето, освен географските локации, треба да се земат во предвид и терестријалните топологии. Иако обезбедуваат важни согледувања во оптимизацијата на сигурност и доцнење, постоечките проблеми за одредувањето на позицијата на контролерот и сателитски gateway имаат едно ограничување: сите тие земаат во предвид само терестријална мрежа.

Целта е да се максимизира доверливоста во комуникацијата од терестријаните свитчеви и од сателитите до контролерите во SDN интегрираната мрежа. Односно, најпрво формално го формулираме проблемот со поставеноста на сателитските *gateways*, со цел да се оптимизира просечното доцнење од сите терестријални свитчеви до сателитот. Потоа, го дефинираме проблемот на заедничко поставување на контролери и gateway-и за да се добие максимална просечна сигурност од терестријалните комутатори и сателити до контролерите со оглед на ограничувањето на доцнењето. Предложен е симулиран алгоритам базиран на загревање (Simulated annealing based algorithm или скратено: SAA) за проблемот со поставувањето на сателитски *gateway*. Со многу помала персметковна комплексност на O(k · n) во секоја итерација, овој алгоритам може да ги постигне речиси оптималните перформанси за доцнење, во спредба со OEA (*Optimal Enumeratio Algorithm*), кој го дава оптималното решение, но со многу голема персметковна комплексност на O(k · n · Ck n). Понатаму се развива уште едно хибридно решение со загревање и кластерирање, односно SACA, за проблемот со заедничката поставеност. SACA може да постигне блиска оптимална доверливост со помало време на извршување O(n^2) во секоја

итерација од OEAJ (optimal enumeration algorithm for joint placement problem), кој може да постигне максимална доверливост, но со многу голема персметковна комплексност на O((k + n) · m $C_n^k \cdot C_n^m$), каде што n, k и m ги претставуваат бројот на терестријални свитчеви, бројот на сателитски gateways и бројот на контролери, соодветно, додека C_n^k и C_n^m се комбинации. Експерименталните резултати покажуваат дека за сите топологии и веројатноста за неуспех.

Во 5G-сателитската интергирана мрежа, поголемиот дел од податочниот сообраќај од терестријалните јазли до сателитот мора да се движи преку сателитски gateways, па според тоа позицијата на сателитскиот gateway мора најпрво да се реши. Досега, ваквиот проблем во интергираната мрежа речиси е недопрен. Постоечките механизми главно се фокусираа на позицијата на gateway во безжичните и сезорски мрежи. За да се постигат најдобрите мрежни перформанси во SDN, многу прашина се крена околу проблемот со позицијата на контролерот. Досега, можеме да заклучиме дека студиите за овој проблем генерално ги користуваат приодите за да ги дознаат најдобрите локации во мрежата каде би ги поставале контролерите со цел да се постигне: минимално доцнење, максимална достапност и минимизирање на трошоците за развој и потрошувачка на енергија. Особено, базирано на целта да се максимизира комуникациската достапност, многу методи се предложени за да се исполнат овие цели.

Архитектура



Сл. 3.21. Пример за SDN сателитска архитектура [36].

Како што е прикажано на сликата 3.21, архитектурата за SDN-овозможената сателитска интегрирана мрежа во 5G е составена од два логички дела, податочна и контролна рамнина.Во податочната рамнина, корисничката опрема се поврзува на 5G мрежата преку gNBs (т.е., gNodeB, 5G NodeB) и RN-и, каде терестријалната 5G мрежа воглавно се ослонува на оптика за да воспостави конекција помеѓу SDN јазли во рамки на позадинската мрежа од базните станици до јадрото (т.е. backhaul-от), како и пристапната радио мрежа т.е. RAN (*Radio Access Network*).

GEO сателитот со висок проток е поставен како составен дел на 5G мрежата во архитектурата, и сателитот комуницира со терестријалните јазли преку сателитски gateway-и и RNs (релејни јазли). Во контролната рамнина, повеќе SDN контролери кои се поставени на физички јазли во податочната рамнина, овозможуваат централизирана контрола и управувачки функции на RAN и на backhaul-от, како и на јадрената мрежа.

Одредување на соодветна позиција на gateway

Во SDN базираната 5G сателитска интегрирана архитектура, најпрвин го разгледуваме процесот за препраќање на сообраќајот од секој терестријален јазел (комутатор) до сателитот преку сателитски gateway. Неопходно е за секој терестријален комутатор да комуницира со сателитот за најрактно можно време. Доцнењето меѓу терестријалниот комутатор и сателитот е значаен фактор кој влијае на перформансите на мрежата. Доцнењето воглавно се состои од две компоненти: пропагациско доцнење и доцнење при рутирање. Пропагациското доцнење се одредува преку растојанието меѓу комутаторот и сателитот, а доцнењето при рутирање зависи од оптовареноста на јазолот. Бидејќи растојанието од сателитот до површината на земјата е скоро константно, доцнењет меѓу комутаторот и сателитскиот gateway е особено важно. Па така, сателитскиот gateway треба да се постави во терестријалната мрежа на позиција со која пропагациското доцнење ќе биде минимално.

Позиционирање на контролери и сателитски *gateways*

Главната функција на контролерот во една ваква архитектура е да прави одлуки за рутирање и спроведување на паметно распределување на сообраќај и диверзитет на јазли. Претпоставуваме дека SDN е овозможен во сателитскиот систем и сателитот е претворен во SDN комутатор, кој прима контролни инструкции од контролер кој е лоциран на земјината површина. Треба да се земе во предвид дека испад на линк

или на јазол може да ги исклучат контролната и податочната рамнина, со што ќе се оневозможи комуникација и примање на инструкции од контролерот и ќе настане голема деградација на сервис и губење на пакети. Па затоа, подобрување на стабилноста на SDN мрежата е од големо значење. Поради влијанието на географските локации, различни поставувања на контролерот може различно да влијаат на стабилност, како и различното позиционирање на сателитите кои играат улога на gateway може да нудат различно доцнење. Па така претпоставувајќи дека секој терестријален комутатор е потенцијална локација за контролер или gateway, наша цела е да поставиме соодветен број на контролери и gateway-и во оптимални локации за да се достигне максимум од интегрираната мрежа базиарана на ограничување од доцнење.

Дефинираме доверливост на пат од секој терестријален комутатор u до контролер c како R_{uc}, и доверливост на пат од сателит s до контролер *c* преку *gateway*-от g како R_{gsc}. Средната доверливост на сите терестријални комутатори и сателите до контролерите, R_{ave} може да биде дефинирана како:

$$R_{ave}=\frac{1}{n+k}\left(\sum_u R_{uc} + \sum_s R_{sc}^{g}\right),\ \ c \in S, g \in W \qquad (3.1)$$

Тука, n и k го покажуваат бројот на терестријални комутатори и сателитски *gateway* -и, додека S го означува сетот на контролери. Претпоставувајќи дека локацијата на секој терестријален комутатор е потенцијален кандидат за да се постави контролер или gateway, наша цел е да се постави доволен број на контролери и gateway-и на оптимални локации за да се постигне максимална средна доверливост на интегрираната мрежа базирано на ограничување на доцнењето.

Позиционирање на сателитски *gateway* за намалување на доцнењето

При формулација на проблемот, мрежата може да се претстави преку граф G(V, E), каде V е множество од јазли (комутатори и сателити), а E претставува сетови од физички линкови меѓу јазлите. Имајќи го смножестото на терестријални јазли $V_T \subset V$, и бројот на сателитски *gateway*-и *k* кои треба да се постават, ние целиме кон пронаоѓање на подмножество од *k* комутатори $W = (g_1,g_2,g_3,\ldots,g_k) \subset V_T$ за да се минимизира просечнато доцнење при поставување k сателитски gateway-и на ($g_1,g_2,g_3,\ldots,g_k$). За комутатор $u \in V_T$ ние треба да пресметаме L_{usg}. Нека L_{ug} го означува доцнењето меѓу терестријалниот комутатор u до gateway $g \in W$ и L_{gs} доцнењето од g до сателит *s*, формулата за одредување на Lusg ќе биде:

$$L_{us}^{g} = L_{ug} + L_{gs} \qquad (3.2)$$

Базирано на овие нотации, може да го формулираме проблемот за поставувањето на *gateway*.

Табела 3.2. Дефиниции и акроними на употребените променливи и величини

Notation	Definition
$G(\mathrm{V}, \mathrm{E})$	Physical network with node set V and link set E, $\mathrm{V} = \mathrm{V}_T \bigcup \{s\}$
V_T	set of terrestrial switch nodes
W	set of satellite gateways
S	set of SDN controllers
u	a switch node in V_T
s	satellite node
g	a satellite gateway in W
c	a controller in S
k	number of satellite gateways
m	number of SDN controllers
P_v	failure probability of terrestrial node
P_e	failure probability of terrestrial link
P_{esg}	failure probability of satellite link from s to g
L_{us}^{g}	latency from u to s via g
L_{ug}	latency from u to g
L_{gs}	latency from g to s
L_{max}	maximum latency the terrestrial network can tolerate
R_{uc}	reliability of the path from u to c
R_{sc}^{g}	reliability of the path from s to c via g

3.4.2 SDN /NFV – базирани терестријални сегменти на сателитски системи

Виртуелизирана сателитска мрежа (*Virtualized Satellite Network* или скратено: VSN) е замислена како сателитска мрежа во која најголемиот дел од функциите се доставени како софтверски компоненти што работаат во една или повеќе *Network Function Virtualization Infrastructure Point of Presence* (NFVIPoP(s)). Обезбедени се не-виртуелизирани функции на Виртуелизирана сателитска мрежа (VSN) преку еден или повеќе сателитски основни gateway-и, кои можат да бидат доделени на даден VSN или споделени меѓу повеќе VSN-и. Операцијата на секој VSN може да биде делегирана на клиентот, дејствувајќи како сателит VNO. Секој VSN може да биде прилагоден на потребите на клиентот, вклучувајќи и различни мрежни услуги кои работат како VNF-и.

Особено, како што е илустрирано на слика 2.15, следните ентитети можат да формираат дел од даден VSN: Една или неколку сателитски мрежни функции VNF-и (SNF-VNF-и) и SBG-VNF-и кои заедно можат да обезбедат функции наVSN;

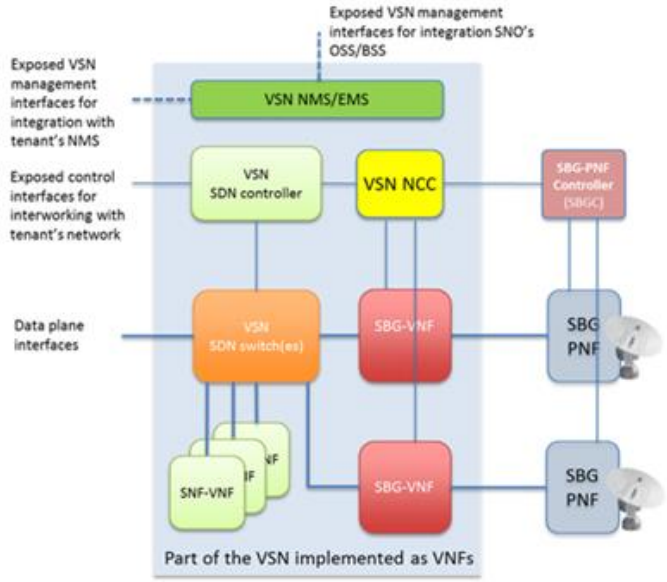
- Контролни апликации и SDN контролери за реализација на некоја VSN контролна функција (управување со радио ресурси, QoS контрола и сл.)
- Управување со мрежата (NM) и управување со функции, исто така работат како VNF-и, кои обезбедуваат паќет на VSN функции на управување (грешка, конфигурација, безбедност итн.)

Друга клучна карактеристика на архитектурата е поддршка за мулти контрола и управување со способности преку изложба на контролни и управувачки интерфејси, како што е покажано на слика 3.22. Таков интерфејс ќе обезбеди унифицирано управување кога VSN меѓусебно работи со терестријални мрежи, на пример од крај до крај сообраќаен инженеринг. Виртуелизација на сателитот GW (Gateway) се изучува идентификувајќи различни варијанти во однос на тоа кои функции можат да бидат импрементирани како виртуелизирани мрежни функции и кои остануваат како спрецијализирани хардверски апарати (на пр. Физички мрежни функции).

Без да биде условен од било која посебна варијанта, во понатамошниот текст го означуваме SNF-VNF до имплементација на SNF како VNF-и, SBG-VNF до имплементација на дел од SBG функциите како VNF и SBGPNF до не-виртуализиран дел од фукциите на SBG. Заедно со претходно дискутираните компоненти, VSN прикажан на слика 2.15 покажува мулти- gateway сателитска мрежа со неколку SBG-VNF и SBG-PNFs заедно со повеќе SNFVNFs. Без губење на општоста, се претпоставува дека постои еден NCC и еден SBG-PNF контролер за неколку порти, иако би биле можни и други опции. Покрај тоа, VSN, исто така се состои од голем број на SDN комутатори и SDN контролери кои можат да бидат виртуелизирани елементи распоредени како дел од VSN. SDN- свитчевите се користат за поддршка на внатрешните L3/L2 препраќачки функции низ компонентите на VSN, како и за меѓусебно поврзување со надворешни мрежи. Како пример, SDN свитчевите можат да додаваат/ испуштаат VLAN тагови /MPLS заглавја, ги спроведуваат различните QoS третмани, итн, и двете за цели за внатрешно или надворешно меѓусебно работење.

VSN SDN контролерот се користи:

1. за контролирање на функциите на SDN свитчевите,
2. да поддржи динамичко активирање на услугите за пренос на податоците преку интерфејс со NCC функциите,
3. да се изложи контролен интерфејс за меѓусебно работење со мрежата на изнајмувачот.



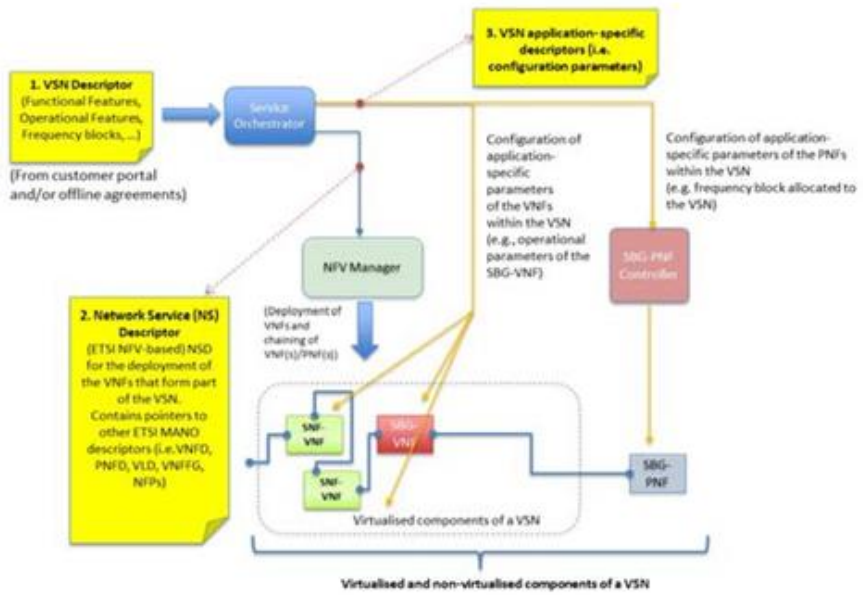
Сл. 3.22. Илустрација на мулти-gateway VSN со вграден SDN контролер што ги изложува контролните интерфејси за меѓусебно работење со мрежата на закупецот [37].

Вреди да се напомене дека слика 3.22 ни дава само илустративни примери, воведувајќи ги главните компоненти на VSN кои подоцна се наведени во описот на работните процеси. Но, други варијанти се можни во зависност од видовите и карактеристиките на VNF-иите кои би можеле да бидат замислени / развиени за да ги задоволат сите специфични потреби (пример, побарувачката на пазарот, приспособливост итн). Контролата и распоредувањето на VSN, управувачките субјекти кои се потребни се прикажани на слика 2.16. Можностите за оркестрација се логички централизирани во така наречена Service Orchestrator (SO) *management component*, која е дел од операциските системи за поддршка(OSS) / бизнис системите за поддршка (BSS) (OSS/BSS) на сателитскиот мрежен оператор (SNO). Покрај тоа, функционалностите поврзани со инстанцијацијата, модификација и терминирање на VNFs кои се дел од еден виртуализиран дел од VSN се опфатени од страна на NFV управувачот.

Функционалностите обезбедени од страна на SO и NFV *Manager* се одговорни за:

- Управување со животниот циклус на VSN, кое може да се дефинира како сет на функции потребни за управување со инстанцијата, оддржување и терминирање на VSN.

- Состав на дескрипторот на мрежни услуги (NSD) кој претставува дел од VSN кој се имплементира како VNFs и е извржен преку NFVI-PoP(и).
- Одредување на аспектите специфични за апликациите на двете VNFs и PNFs кои се дел од VSN.
- Дефект, конфигурација, изведба, безбедносно (FCAPS) управување со VSN и неговите компоненти, без оглед дали се VNF-и или PNF-и. Овие FCAPS управувачки функционалности поддржани од SO главно се наменети да гарантираат правилна операција на функционирање(пример, ракување со аларми, корекција на проблем, употреба на ресурси итн).
- Управување со животниот циклус на мрежната услуга(NS) на VSN преку интеракција со NFV менаџер, кој ја нуди Os-Ma-nfvo референтна точка како што е наведено во ETSI NFV MANO архитектурата. Вреди да се напомене дека NFV менаџерот води грижа за специфичната конфигурација на VNFs кои се дел од NS.
- Управувањето со VNF паќетите може да биде веќе вклучено на NFV менаџер или може да биде менаџирано/вклучено на контролна табла на SO.Во принцип, кај SO само VNFDs се доволни за составување на NSDs.



Сл. 3.23. Илустрација на VSN и компонентите за оркестрација / управување [38].

3.5 Заклучок

Современите комуникациски и информациски технологии, како со 5G, така и со IoT, а несомнено имплементирани во паметните градови ќе осигураат поквалитетен живот на луѓето, а со оглед на тоа дека се поголемиот дел од населението на планетава заминува да живее во градовите, ќе придонесе за намалување на товарот и несаканите последници од пренаселеноста и загадување во истите. Така, паметните градови се иднина и поединечни проекти кои го поддржуваат нивниот развој посочуваат на потенцијални проблеми при нивна реализација, односно на потребната инфраструктура и на самото население. За да се овозможи тоа да функционира беспрекорно, несомнено во самата комуникациски инфраструктура лежи мрежна виртуелизација како неделив процес на комбинирање на хардверски и софтверски мрежни ресурси и мрежна функционалност во единствена рамка.

Во рамките на оваа глава беа разгледани *Software-defined networking* (SDN) и Network Function Virtualization (NFV) кои заедно ја формираат мрежната виртуализација, и практично како релативно нови пристапи кон дизајнот на мрежите и функционирањето на истите, се една од најистражуваните и најприменуваните теми во денешницата.

Воедно, **предвидено е дека NFV + SDN, заедно со пресметките во облак, ќе стане критична овозможувачка технологија за радикално да го револуционизира начинот** на кој мрежните оператори ќе го создадат, дизајнираат и ќе ја монетизираат нивната инфраструктура.

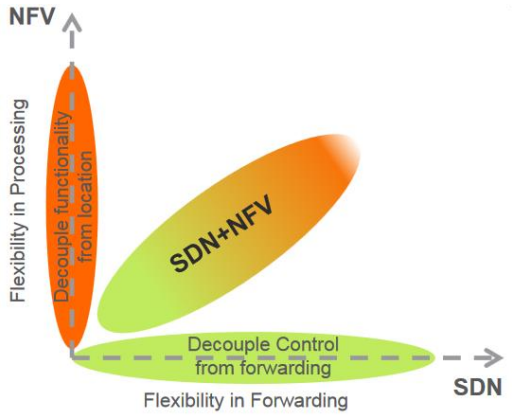
Така, SDN всупност го централизира и автоматизира управувањето со мрежните уреди, воведувајќи сосдветна апстракција на физичките ресурси во мрежата. При тоа, SDN дозволува да се одделат податоците и контролните планови, а со тоа да бидат полесни за подобрувања и управување.

Од друга страна, NFV е проспективно обединувачка револуција, нудејќи повеќе можности за приходи во синџирот на вредност.

Заедно, и двете SDN и NFV - се тесно поврзани, но не се заменливи и не се комплетно еквивалентни. Како што беше опишано во подглавите - тие се различни, но многу комплементарни технологии кои можат да се користат, или сами засебно или во комбинација. NFV може да се имплементира без да се бара SDN, и обратно - мрежните оператори можат да изградат SDN без NFV.

Сепак, и двата пристапи може да се комбинираат и ова комбинацијата нуди силна синергија за мрежните оператори и

провајдери на Интернет услуги, и се еден од носечките столбови во идните 5G мобилни и безжични мрежи (види Слика 3.24).



Слика 3.24. Илустрација на флексибилноста на SDN + NFV во 5G [39].

На крајот од оваа глава беа дадени некои примери за примена на мрежната виртуализација (SDN и NFV) токму во најновите истражувања и архитектури на паметни градови, IoT, 5G и сателитски мрежи.

Развојот на паметни градови со сите технолошки подобрувања, особено преку примена на информатичка и комуникциска технологија и мрежната виртуализација, мора во поголема мера да биде отворен за социјална, емоционална и духовна страна на животот на човекот. Повеќе од половина од населението на Земјата живее во градови кои искористуваат над 80% од расположливите ресурси. Современите градови претставуваат огледало на развојот на нашата цивилизација кои како силни магнети привлекуваат најквалитетни човечки, технолошки, организациски и природни ресурси. Паметните градови треба да се развиваат кон интегрирани, односно целосни градови во кои свеста за поврзаноста и меѓусебната условеност е применета во сите области на човечкиот живот и неговиот однос со природата која го опкружува.

На крај, мрежната виртуализација (NFV + SDN), заедно со сервисите на IoT, мобилните мрежи 5G, пресметките во облак и другите технологии и сервиси кои се појавуваат, ќе стане критична овозможувачка технологија за радикално да го промени начинот на живот на сите, во насока на обезбедување висок квалитет на услуги, сервиси и несомнено ќе даде јасна подлога за развој на многу интелигентни платформи и паметни нешта.

Користена литература

[1] S. Ortiz, "*Software-Defined Networking: On the Verge of a Breakthrough?*" Computer, vol. 46, no. 7, pp. 10-12, 2013.

[2] C. S. Li and W. Liao, "*Software Defined Networks*", Editorial, IEEE Comm. Mag., Feb. 2013.

[3] M. Casado, T. Koponen, S. Shenker, and A. Tootoonchian, "*Fabric: a retrospective on evolving SDN*," in Proc. of Workshop on Hot Topics in Software Defined Networking, Aug. 2012, pp. 85-90.

[4] Y. Kanaumi, S. Saito, and E. Kawai, "*Toward large-scale programmable metworks: lessons learned through the operation and management of a wide-area Openflow-based network*" in Proc. of Int. Conf. on Network and Services Management, Oct. 2010, pp. 330-333.

[5] H. Fei, *Network Innovation through OpenFlow and SDN: Principles and Design*, Taylor & Francis LLC, CRC Press, to appear in 2014.

[6] B. Lantz, B. Heller, and N. McKeown, "*A network in a laptop: rapid prototyping for software-defined networks*," in Proc. of ACM SIGCOMM Workshop on Hot Topics in Networks, New York, NY, USA, 2010, pp. 19:1-6.

[7] T. D. Nadeau and P. Pan, "*Software Driven Networks Problem Statement*," IETF Internet-Draft (work-in-progress), draft-nadeau-sdnproblem-statement-01, Oct. 2011.

[8] M. Yu, J. Rexford, M. Freedman, and J. Wang, "*Scalable flow-based networking with difane*," ACM SIGCOMM Computer Communication Review, vol. 41, pp. 351-362, 2010.

[9] K. Yap, M. Kobayashi, R. Sherwood, T. Huang, M. Chan, N. Handigol, and N. McKeown, "*Openroads: Empowering research in mobile networks*," ACM SIGCOMM Computer Communication Review, vol. 40, no. 1, pp. 125-126, 2010.

[10] ITU-T Recommendation Y.3300, "*Framework of software-defined networking*", June 2014.

[11] OpenFlow Switch Specification: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf

[12] Open Networking Foundation, https://www.opennetworking.org/

[13] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "*Openflow: enabling innovation*

in campus networks," ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, pp. 69-74, 2008

[14] R. Sherwood, M. Chan, and A. Covington, "*Carving research slices out of your production networks with openflow*," ACM SIGCOMM Computer Communication Review, vol. 40, no. 1, pp. 129-130, 2010.

[15] P. Dely, A. Kassler, and N. Bayer, "*Openflow for wireless mesh networks*," in Proc. of Int. Conf. on Computer Communications and Networks, pp. 1-6, 2011.

[16] Pica8OpenNetworkFabric, http://www.pica8.org/solutions/openflow.php.

[17] Indigo − Open Source OpenFlow Switches, http://www.openflowhub.org/display/Indigo/.

[18] P. T. Congdon, P. Mohapatra, M. Farrens, and V. Akella, "*Simultaneously reducing latency and power consumption in OpenFlow switches*," IEEE/ACM Trans. on Networking, vol. PP, no. 99, pp. 1-14, 2013.

[19] A. Khan and N. Dave, "*Enabling hardware exploration in softwaredefined networking: a flexible, portable OpenFlow switch*," in Proc. of Annual Int. Symp. on Field-Programmable Custom Computing Machines, 2013, pp. 145-148.

[20] K. Bakshi, "*Considerations for Software Defined Networking (SDN): Approaches and use cases*," in Proc. of IEEE Aerospace Conf., 2013 , pp. 1-9.

[21] F. Hu, Q. Hao and K. Bao, "*A Survey on Software-Defined Network and OpenFlow: From Concept to Implementation*," in IEEE Communications Surveys & Tutorials, vol. 16, no. 4, pp. 2181-2206, Fourthquarter 2014. doi: 10.1109/COMST.2014.2326417

[22] Biswas, A. A. Lazar, J. F. Huard, K. S. Lim, S. Mahjoub, L. F. Pau, M. Suzuki, S. Torstensson, W. Wang, and S. Weinstein, "*The IEEE P1520 Standards Initiative for Programmable Network Interfaces*," IEEE Commun. Mag.,Vol. 36, no. 10, 1998, pp. 6470.

[23] A. Doria, J. H. Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, and J. Halpern, "*Forwarding and Control Element Separation (ForCES) Protocol Specification*," in [Online]. Available: http://tools.ietf.org/html/rfc5810

[24] T. V. Lakshman, T. Nandagopal, R. Ramjee, K. Sabnani, and T. Woo, "*The SoftRouter Architecture*," in Proc. ACM SIGCOMM Workshop on Hot Topics in Networking, 2004.

[25] ITU-T Recommendation X.1038 (10/2016): *Security requirements and reference architecture for software-defined networking*.

[26] Leng, J., Zhou, Y., Zhang, J., and Hu, C. (2015), *An Inference Attack Model for Flow Table Capacity and Usage: Exploiting the Vulnerability of Flow Table Overflow in Software-Defined Network*, arXiv:1504.03095.

[27] ETSI, "*Network Functions Virtualisation (NFV); Architectural Framework*," GS NFV 002 (v. 1.1.1), Oct. 2013.

[28] ITU-T Recommendation Y.3011, "*Framework of network virtualization for future networks*", 2012.

[29] ITU-T Recommendation Y.3012, "*Requirements of network virtualization for future networks*", 2014.

[30] https://medium.com/all-technology-feeds/smart-connected-and-iot-based-devices-whats-the-difference-36fc1bdc36b2

[31] http://www.interreg-danube.eu/uploads/media/approved_project_public/0001/03/8e42ee2ff f53ba9eda964995fa6dddc4fd564998.pdf

[32] Deloitte, "*Smart Cities: The importanceof a smart ICT infrastructure forsmart cities*",

link:https://www.stokab.se/Documents/Nyheter%20bilagor/SmartCityInfraEn.pdf

[33] P. Demestichas, A. Georgakopoulos, D. Karvounas, K. Tsagkaris, V. Stavroulaki, J. Lu, C. Xiong, and J. Yao, "*5G on the Horizon: Key Challenges for the Radio-Access Network*," VT Magz, IEEE, vol. 8, pp. 47–53, Sept 2013.

[34] *View on 5G Architecture* - 5G PPP Architecture Working Group (2016), link: https://bscw.5g-mmmagic.eu/pub/bscw.cgi/191293

[35] Jinzhen Bao, Baokang Zhao, Wanrong Yu, Zhenqian Feng, Chunqing Wu, and Zhenghu Gong. 2014. *OpenSAN: a software-defined satellite network architecture*. In Proceedings of the 2014 ACM conference on SIGCOMM (SIGCOMM '14). ACM, New York, NY, USA, 347-348.

[36] J. Liu, Y. Shi, L. Zhao, Y. Cao, W. Sun and N. Kato, "*Joint Placement of Controllers and Gateways in SDN-Enabled 5G-Satellite Integrated*

Network," in IEEE Journal on Selected Areas in Communications, vol. 36, no. 2, pp. 221-232, Feb. 2018.

[37] R. Ferrus, O. Sallent, T. Ahmed and R. Fedrizzi, "*Towards SDN/NFV-enabled satellite ground segment systems: End-to-End Traffic Engineering use case*," 2017 IEEE International Conference on Communications Workshops (ICC Workshops), Paris, 2017, pp. 888-893.

[38] T. Ahmed, R. Ferrus, R. Fedrizzi and O. Sallent, "*Towards SDN/NFV-enabled satellite ground segment systems: Bandwidth on Demand use case*," 2017 IEEE International Conference on Communications Workshops (ICC Workshops), Paris, 2017, pp. 894-899.

[39] http://www.itc26.org/fileadmin/ITC26_files/ITC26-Tutorial-Rostami.pdf

[40] https://ec.europa.eu/digital-single-market/smart-cities

[41] https://www.digitale-technologien.de/DT/Navigation/EN/Foerderprogramme/Smart_Service_Welt/smart_service_welt.html

[42] https://www.gemalto.com/mobile/inspired/5G

[43] K. Wang et al., "*Wireless Big Data Computing in Smart Grid*," IEEE Wireless Commun., vol. 24, no. 2, Apr. 2017, pp. 58–64.

[44] K. Wang et al., "Mobile Big Data Fault-Tolerant Processing for eHealth Networks," IEEE Network, vol. 30, no. 1, Jan./ Feb. 2016, pp. 36–42.

[45] http://www.esru.strath.ac.uk/EandE/Web_sites/12-13/SmartCities/smartcities.html

[46] http://www.smartecocity.com/category/singapore-2/

[47] http://www.pmo.gov.sg/smartnation

[48] http://ict.telekom.mk/smart-city.nspx

[49] http://www.poslovni.hr/tehnologija/drive-me-projekt-autonomne-voznje-334074

[50] B. G. Evans, "*The role of satellites in 5G*," 2014 7th Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications Workshop (ASMS/SPSC), Livorno, 2014, pp. 197-202. doi: 10.1109/ASMS-SPSC.2014.6934544

[51] "*Scenarios, requirements and KPI's for 5G mobile and wireless systems*"ICT-317669-METIS/D1.1.April2013. www.metis2020.com/docum,ents/deliverables

[52] ETSI GS NFV 001 V1.1.1, "*Network Functions Virtualisation (NFV): Use Cases*," Oct. 2013.

[53] ETSI TS 101 545-1 V1.1.1, "*Digital Video Broadcasting (DVB);. Second Generation DVB Interactive Satellite System (DVB-RCS2), Part 1: Overview and System Level specification*," May 2012.

[54] ETSI TS 102 672, "*Satellite Earth Stations and Systems (SES); Broadband Satellite Multimedia (BSM); Management Functional Architecture*", November 2009.

[55] ETSI TS 102 673, "*Satellite Earth Stations and Systems (SES); Broadband Satellite Multimedia (BSM); Performance Parameters*", November 2009.

[56] Fenech, H.; Amos, S.; Tomatis, A.; Soumpholphakdy, V., "*High throughput satellite systems: An analytical approach*," IEEE Transactions on in Aerospace and Electronic Systems, January 2015

[57] Bertaux L., Medjiah S., Berthou P., Abdellatif S., Hakiri A., Gelard P., "*Software Defined Networking and Virtualization for Broadband Satellite Networks*". IEEE Communications Magazine, March 2015

[58] Sacchi, C.; Bhasin, K.; Kadowaki, N.; Vong, F., "*Toward the "space 2.0" Era [Guest Editorial]*," Communications Magazine, IEEE, vol.53, no.3, pp.16,17, March 2015

[59] B. Aoun, R. Boutaba, Y. Iraqi, and G. Kenward, "*Gateway placement optimization in wireless mesh networks with QoS constraints*," IEEE J. Sel. Areas Commun., vol. 24, no. 11, pp. 2127–2136, Nov. 2006.

[60] U. Ashraf, "*Energy-aware gateway placement in green wireless mesh networks*," IEEE Commun. Lett., vol. 21, no. 1, pp. 156–159, Jan. 2017.

[61] D. Hock, S. Gebert, M. Hartmann et al., "*Pareto-optimal resilient controller placement in SDN-based core networks*," in IEEE ITC, 2013, pp. 1–9.

[62] M. T. I. ul Huque, G. Jourjon, and V. Gramoli, "*Revisiting the controller placement problem*," in IEEE LCN, 2015, pp. 450–453.

[63] A. Sallahi and M. St-Hilaire, "*Optimal model for the controller placement problem in Software Defined Networks*," IEEE Commun. Lett., vol. 19, no. 1, pp. 30–33, Jan. 2015.

[64] Marcus, J. Scott, Molnar, Gabor, "*Network sharing and 5G in Europe: The potential benefits of using SDN or NFV*", International Telecommunications Society (ITS): "*Competition and Regulation in the Information Age*", Passau, Germany, 2017

[65] T. Helaly, N. Adnani "*A fourth category of software-defined instrumentation for wireless test*", IEEE Instrumentation & Measurement Magazine, Vol. 20, Iss. 4, Aug. 2017

[66] G. Jue and S. Ferguson, "*RF And Digital Tests Unite Against BER*", Wireless Systems Design Magazine, Nov. 2004

[67] "*Connected Simulation and Test Solutions Using the Advanced Design System*", Agilent Technologies, Application Note Number 1394

[68] G. Jue, "*3GPP W-CDMA Systems: Design and Test*", IEEE Microwave Magazine, June 2002, pp 56-64

[69] B. Zarlingo, K. Kalbasi and G. Jue, "*Flexible Digital Demodulation/Integrating Simulation Software with Measurement Hardware*", IEEE Autotestcon 2002

[70] D. Leiss, "*Combined Virtual and Physical Hardware Performance Analysis*" IEEE Autotestcon 2002