საქართველოს ტექნიკური უნივერსიტეტი

ალექსანდრე ზანგალაძე

უსადენო კავშირის ტექნოლოგიების საგნების ინტერნეტისთვის გამოყენების პრობლემების კვლევა

წარმოდგენილია დოქტორის აკადემიური ხარისხის მოსაპოვებლად

სადოქტორო პროგრამა: ციფრული სატელეკომუნიკაციო ტექნოლოგიები

შიფრი: 0714

საქართველოს ტექნიკური უნივერსიტეტი

თბილისი, 0166, საქართველო

ივლისი, 2021 წელი

თბილისი

2021 წელი

საქართველოს ტექნიკური უნივერსიტეტი

ენერგეტიკისა და ტელეკომუნიკაციის ფაკულტეტი

ჩვენ, ქვემორე ხელისმომწერნი ვადასტურებთ, რომ გავეცანით ალექსანდრე ზანგალაძე მიერ შესრულებულ სადისერტაციო ნაშრომს დასახელებით: **“უსადენო კავშირის ტექნოლოგიების საგნების ინტერნეტისთვის გამოყენების პრობლემების კვლევა“** და ვაძლევთ რეკომენდაციას საქართველოს ტექნიკური უნივერსიტეტის ენერგეტიკისა და ტელეკომუნიკაციის ფაკულტეტის სადისერატაციო საბჭოში მის განხილვას დოქტორის აკადემიური ხარისხის მოსაპოვებლად.

....., 2021 წელი

ხელმძღვანელი ---------- ასოც. პროფესორი შ. კვირკველია

რეცენზენტი ----------

რეცენზენტი ----------

საქართველოს ტექნიკური უნივერსიტეტი

2021

ავტორი: ალექსანდრე ზანგალაძე

თემის დასახელება: „უსადენო კავშირის ტექნოლოგიების საგნების ინტერნეტისთვის გამოყენების პრობლემების კვლევა“

ფაკულტეტი: ენერგეტიკისა და ტელეკომუნიკაციის

სადოქტორო პროგრამა: „ციფრული სატელეკომუნიკაციო ტექნოლოგიები“

ხარისხი: აკადემიური დოქტორი

სხდომა ჩატარდა: ივლისი, 2021 წელი

რეზიუმე

სადღეისოდ, ნივთების ინტერნეტში აქტუალურ საკითხად ითვლება სენსორების უწყვეტი მუშაობა, მცირე დაყოვნება და დაბალი ენერგომოხმარება. აღსანიშნავია, რომ ამ სენსორებს არ აქვთ წვდომა ენერგიის გარე წყაროსთან და უწევთ ენერგიის მიღება მცირე ზომის ელემენტებიდან. სენსორები ძირითად განლაგებულია ინფრასტრუქტურისგან მოშორებულ ადგილებზე, სადაც მონაცემების გადაცემას ახდენენ რადიოქსელების მეშვეობით. რადიოქსელის ტევადობა მნიშვნელოვანია მასში ჩართული მოწყობილობების ოპერატიული მუშაობისთვის. როდესაც სენსორებს უწევთ რადიოქსელში მონაცემების პერიოდულად გადაცემა, ისინი ზრდიან ქსელის დატვირთვას. გარდა ამისა, რადიოსიგნალის გადაცემა მოითხოვს მნიშვნელოვნად დიდ ენერგიას, რაც ამცირებს ბატარეიის ექსპლუატაციის დროს.

მონაცემების დიდი ნაწილი არის პსევდო სტატიკურ ინფორმაციაზე ორიენტირებული. ინტერნეტში ასეთი ინფორმაცია მრავალჯერ და პერიოდულად გადის ერთსა და იმავე გზას, რადგან მოთხოვნილია მრავალი მომხმარებლის მიერ. ამ მონაცემების გადაცემა ხდება TCP/IP პროტოკოლთა კრებულის საშუალებით. მასში არსებული პროტოკოლების მიზანია მონაცემთა გადაცემა ინფორმაციის ორიგინალურ წყაროსა და მიმღებს შორის. აქამდე, OSI მოდელის ზედა დონეების შრეებზე მომუშავე ტექნოლოგიებით რადიოარხების განტვირთვა არ ხდებოდა. დღეს, უკვე ყალიბდება ახალი ქსელების კონცეფციები. ასეთები არის ინფორმაციაზე ორიენტირებული ქსელები. ამ ქსელების საშუალებით ქსელურ კვანძებზე შესაძლებელია განმეორებითი მონაცემების გადაცემის შემცირება. დისერტაციაში შემოთავაზებულია ქსელის მოდელი, რომელიც უზრუნველყოფს სენსორების პერიოდული ტრაფიკის გადაცემის შემცირებას. ამის ხარჯზე დაიზოგება ენერგია და შემცირდება რადიოარხების დატვირთვა. ჩემს მიერ მოყვანილ იქნა ICN ტიპის ქსელი, კერძოდ NDN-ის სტრუქტურა და გამოყენების სპეციფიკა IoT-ის ქსელში. განხილულია თავსებადობის და უსაფრთხოების საკითხები არსებულ ქსელთან.

NDN ქსელი მოისაზრებს OSI მოდელის მესამე დონეს. ის არის IPv4/IPv6-ის შემცვლელი პროტოკოლი. სადღეისოდ გლობალურ ქსელში თითქმის ყველა მოწყობილობა მუშაობს IPv4/IPv6-ის პროტოკოლების საშუალებით. NDN ქსელის არსებულ ქსელში პირდაპირი ინტეგრაცია ვერ მოხერხდება განსხავებული არქიტექტურის გამო. ყველაზე დიდ პრობლემას წარმოადგენს ის, რომ არსებულ ქსელის ელემენტებს არ ესმით NDN-ის სტრუქტურა და არსებული ქსელის შეცვლა დაკავშირებულია დიდ ხარჯებთან. NDN-ის გამართული მუშაობა დამოკიდებულია არა მარტო პროგრამულ უზრუნველყოფაზე არამედ ქსელის ელემენტების ფიზიკურ შესაძლებლობებზე. რადგან ქსელში არსებული ელემენტების შეცვლა წარმოადგენს სირთულეს ამ ეტაპზე შეგვიძლია NDN-სა და IPv4-ის

კომბინირებული ვარიანტის გაკეთება. წესით და რიგით მომავალში NDN პაკეტები ენკაფსულირებული იქნება Layer 2-ის დონეზე, მაგრამ ჩვენ ექსპერიმენტებში გამოვიყენეთ Layer 3-ის ენკაფსულაცია.

თბილისის გარკვეულ უბანზე ავაგეთ და გავტესტეთ სატესტო NDN ქსელი. ექსპერემენტის ჩასატარებლად გამოყენებული იყო ჩვეულებრივი კომპიუტერები და Linux როუტერები, სადაც დაყენებული გვქონდა NFD ფრეიმვორკი. მონაცემები ინკაპსულირებული იყო ქართული ინტერნეტ პროვაიდერების IPv4 ქსელებში. ერთ-ერთ მოწყობილობაზე, კერძოდ სერვერზე წინასწარ იყო დაგენერირებული NDN-ის კონტენტი. სერვერზე გარკვეული პერიოდებით შემოდიოდა ერთდროულად როგორც NDN ასევე IPv4-ის მოთხოვნები და მას უნდა დაეკმაყოფილებინა ეს მოთხოვნები. როგორც შედეგებმა აჩვენეს NDN-ის შემთხვევაში დაყოვნება საგრძნობლად შემცირდა. NDN-ი ემსახურება რამდენიმე მიზანს: ის ზედმეტად არ ტვირთავს ძირითად სერვერს და ამავე დროულად ემსახურება ლოკალურად გარკვეული რაოდენობის ქსელის მომხმარებლებს. NDN-ის პლატფორმა სერვერს ართმევს გარკვეული რაოდენობის ინფორმაციულ ტრაფიკს და შესაბამისად განტვირთვას მას. ასევე, IoT-ის მოწყობილობებზე მცირდება ენერგომოხმარება და ქსელში გაგზავნილი პაკეტების დაყოვნების დრო. NDN-ში გამოიყენება დაცულობის მექანიზმები და მათი გამოყენება IPv4-ის მექანიზმებთან ერთად საბოლოო ჯამში უფრო საიმედოს და ეფექტურს გახდის ჰეტეროგენული ქსელით ინფორმაციის გადაცემას. MiniNDN პლატფორმაზე შევაფასეთ NDN-ის და IPv4-ის დაყოვნებები და შედეგად მივიღეთ, რომ დაყოვნება შემცირდა საშუალოდ 60 %-ით. აღსანიშნავია, რომ თავდაპირველად NDN-ის შემთხვევაში დაყოვნება გაიზარდა 17%-ით, ვინაიდან პირველი პაკეტის დაქეშირების დროს სიგნალის დამუშავებას დაჭირდა გარკვეული პერიოდი. დისერტაციაში შევაფასეთ NDN IoT და IPv4 IoT ქსელები და მოვახდინეთ მათი შედარება. ნაშრომში გამოყენებულია NFD ფრეიმვორკი და აგრეთვე სენსორები, რომლებიც იყო დაწერილი პითონის პროგრამულ ენაზე. სენსორებზე ენერგომოხმარების დაზოგვა დამოკიდებულია მასზე მოსული მოთხოვნების რაოდენობაზე. ერთ-ერთ ჩატარებულ ექსპერიმენტში ენერგომოხმარება დაეცა 23,7%-ით. აღსანიშნავია ისიც, რომ რადიოკვანძების დატვირთვაც ზოგადად შემცირდა. ენერგომოხმარების მნიშვნელობა გამოვხატეთ ფორმულით.

Abstract

Nowadays, question of present interest are: continuous operation of sensors as well as small latency and low power consumption. It is noteworthy that these sensors do not have access to an external power source and have to receive power from small batteries. The sensors are mainly located in places far from the infrastructure, where data is transmitted by using radio networks. It is known, that sensors have to periodically transmit data over the radio network that is why they increase the network loading. In addition, transmitting a radio signal requires significantly more power, which reduces battery life generally. Much of the data is focused on pseudo-static information. Such information on the Internet goes through the same path many times and periodically as it is requested by many users. This data is transmitted through a set of TCP / IP protocols. The purpose of the aforementioned protocols is to transfer data between the original source of information and the recipient.

Until now, radio channels with technologies working on the upper levels of the OSI model have not been unloaded. Today, new networking concepts are being formed. These are information-oriented networks. By using these networks, it is possible to reduce duplicate data transmission over network nodes. My dissertation proposes a network model that provides a reduction in the transmission of periodic informational traffic from sensors. This will save energy and reduce the loading on radio channels. I have introduced the ICN type network, in particular the structure of the NDN and the specifics of its use in the IoT network as well as compatibility and security issues with an existing network are discussed. The NDN network assumes the third level of the OSI model. It is an IPv4 / IPv6 replacement protocol. Today, almost every device on the World Wide Web operates via IPv4 / IPv6 protocols. Direct integration into the existing NDN network is not possible due to different architecture. The biggest problem is that existing network elements do not understand the structure of the NDN and modifying an existing network is associated with high costs. The proper operation of the NDN depends not only on the software but also on the physical capabilities of the network elements. Because changing the elements in the network is so difficult by virtue of the above, that is why at this point we can make a combined version of NDN and IPv4.

As a rule, NDN packets will be encapsulated at Layer 2 levels in the future, but we used Layer 3 encapsulation in our experiments. We built and tested NDN network at a specific area of Tbilisi. computers and Linux routers were used to conduct the experiment, where we had the NFD framework installed. The data were encapsulated in the IPv4 networks of Georgian Internet providers. NDN content was pre-generated on one of the devices, namely the server. Both NDN and IPv4 packets were received on the server at regular intervals, and server had to reply to these requests. As the results showed in the case of NDN, the delay was significantly reduced. NDN serves several purposes: it does not overload the main

server and at the same time serves a certain number of local network users. The NDN platform deprives the server of a certain amount of information traffic and therefore unloads it. Also, IoT devices reduce power consumption and latency of packets sent to the network. NDN uses security mechanisms and their use in conjunction with IPv4 mechanisms will ultimately make the transmission of information over a heterogeneous network more reliable and efficient. On the MiniNDN platform, we estimated the latencies of NDN and IPv4 and found, that the latency was reduced by an average of 60%. It is noteworthy that initially, in the case of NDN, the delay was increased by 17% since the signal processing during the first packet caching took some time. In the dissertation we evaluated NDN IoT and IPv4 IoT networks and compared them. In the dissertation the NFD framework was used, as well as sensors that were written in the Python programming language. Saving energy consumption on sensors depends on the number of demands on it. In one of the experiments conducted, energy consumption fell by 23.7%. It should also be noted that the load on the radio nodes was also generally reduced. We expressed the importance of energy consumption by the formula.

სარჩევი

ცხრილების ნუსხა

ნახაზების ნუსხა

გამოყენებული აბრევიატურების ნუსხა

ITU	International Telecommunication Union	საერთაშორისო სატელეკომუნიკაციო კავშირი
ICN	Information Centric Networking	ინფორმაციაზე ორიენტირებული ქსელი
IMT 2020	International Mobile Telecommunication 2020	საერთაშორისო მობილური კავშირი 2020
NDN	Named Data Networking	სახელდარქმეულ მონაცემთა ქსელი
LTE	Long Term Evolution	ხანგრძლივი განვითარების პროექტი
EPC	Evolved Packet Core	განვითარებადი პაკეტთა ბირთვი
TCP/IP	Transmission Control Protocol/IP Protocol	გადაცემის კონტროლის პროტოკოლი
PIT	Pending Interest Table	მოლოდინის ინტერესთა ცხრილი
FIB	Forwarding Information Base	ინფორმაციის გადაცემის ბაზა
CS	Content Store	კონტენტის საცავი
SSD	Solid-State Drive	სოლიდური მყარი დისკი
DNS	Domain Name System	დომენის სახელების სისტემა
OSI	Open Systems Interconnection model	ღია სისტემების ურთიერთკავშირის მოდელი
NDNS	A DNS-Like Name Service for NDN	დომენის სახელების სისტემა სახელდარქმეულ მონაცემთა ქსელისთვის
NFC	Near-field communication	ახლო დისტანციის რადიოსიხშირული იდენტიფიკაციის გაფართოება
LoRa	Long Range	გრძელი დიაპაზონი
LPWAN	A low-power wide-area network	დაბალ ენერგიაზე მომუშავე ფართო ქსელი

NB LTE	NarrowBand LTE	ვიწროზოლოვანი LTE
5G	The fifth generation	მეხუთე თაობა
WCDMA	Wideband Code Division Multiple Access	ფართოზოლოვანი კოდური დაყოფის შეღწევა
LTN	Low throughput networks	დაბალი გამტარუნარიანობის ქსელები
WLAN	Wireless Local Area Network	უსადენო ლოკალური ქსელი
BLE	Bluetooth Low Energy	ბლუთუსი დაბალი ენერგიის მოხმარებით
WiFi	Wireless Fidelity	ერთგული უსადენო ქსელი

შესავალი

პრობლემის აქტუალურობა. სადღეისოდ, IoT-ის ქსელში დაყოვნების შემცირება, ენერგომოხმარების დაზოგვა, ზოგადად ქსელის უბნებზე დატვირთვის შემცირება და დამაბოლოებელ მოწყობილობების განტვირთვა ითვლება კრიტიკულ საკითხებად. საგნების ინტერნეტის ფართო რეალიზაცია დამოკიდებულია რადიოსისტემების გამტარუნარიანობაზე და მის დატვირთვაზე. სისტემის ტევადობა გადანაწილებულია მოწყობილობებს შორის. ამჯერად ცალკეული მოწყობილობების ტრაფიკი მცირეა, თუმცა სისტემაში ათასობით სენსორის დაერთების შემთხვევაში ახალი თაობის მობილურ სისტემებს გაუჭირდებათ მუშაობა მაღალი წარმადობით. გარდა ამისა, ავტომატიზირებული მანქანებისა და დრონების სრულყოფილი მუშაობისთვის საჭირო იქნება სისტემის არაგადატვირთულ მდგომარეობაში ფუნქციონირება. კლასიკურ ინტერნეტის ქსელებში ყველა მონაცემი გადის გზას საწყისი მონაცემების წყაროსა და მოთხოვნს შირის. საწყისი წყარო შეიძლება იყოს სენსორი, რომელიც პერიოდულად აკეთებს გაზომვებს. თუ სენსორზე მოსული მოთხოვნების პერიოდი ნაკლებია სენსორის გაზომვის პერიოდზე, ერთი და იგივე მონაცემები იქნება გაგზავნილი სენსორის რადიოსისტემის არხში. ასეთი მუშაობის ტიპი არაეფექტურია. გარდა ამისა, სენსორის ელემენტის ექსპლუატაციის ვადა მცირდება. აქამდე, რადიოსისტემების ეფექტურობის გაზრდას ცდილობდნენ უშუალოდ OSI ფიზიკური შრეზე და არ იყო განხილული სხვა შრეებით ამ სისტემების გაუმჯობესება.

სამუშაოს მიზანი და კვლევის ამოცანები: ჩვენი სამუშაოს მიზანია IoT-ის ქსელში ისეთი პრობლემების გამოკვლევა როგორიც არის: დაყოვნების შემცირება, ენერგომოხმარების დაზოგვა, ზოგადად ქსელის უბნებზე დატვირთვის შემცირება და დამაბოლოებელი მოწყობილობების განტვირთვა. პრობლემების კვლევა და მათი შეფასება. მომავლის

ინტერნეტის ქსელის გამოყენებით მათი შედარება არსებულ IPv4 IoT-ის ქსელთან.

კვლევის მეთოდოლოგია: NDN ქსელის ვირტუალიზაციის საშუალებას გვაძლევს MiniNDN-ი. ზემოთ აღნიშნული ინსტრუმენტარი წარმოადგენს უფასო პროგრამულ უზურნველყოფას, რომელიც აგებულია Linux-ის და მისი მონათესავე ოპერატიული სისტემების ბაზაზე. MiniNDN-ი ინტერნეტში ყველასთვის ხელმისაწვდომია. იგი თავსებადია მრავალ პროგრამულ ენასთან როგორიცაა: C++, Java, Javascript, Python, .NET Framework(C#). ჩვენი მიზნის მისაღწევად გამოვიყენეთ:

1) მომავლის ინტერნეტის ქსელი, რომელიც აგებულია ICN პრინციპზე. სწორედ ამიტომ ჩვენს მიერ იყო შესწავლილი და გაანალიზებული NDN-ის ქსელის სპეციფიკა და სტრუქტურა თეორიულად.
2) Python-ის პროგრამული ენა, რომლის საშუალებით მოვახდინეთ სენსორების სიმულაცია MiniNDN-ის პლატფორმაზე.
3) Linux მოწყობილობებზე NFD პლატფორმა რათა აგვეგო NDN-ის სატესტო ქსელი. თავდაპირველად MiniNDN-ის პროგრამული ინსტრუმენტარის საშუალებით მოვახდინეთ NDN ქსელის სიმულაცია, ხოლო პითონის საშუალებით კი განვახორციელეთ ქსელში ჩართული სენსორების სიმულაცია.

მეცნიერული სიახლე: მოგეხსენებათ, რომ IoT-ის მოწყობილობებზე კრიტიკულ საკითხად კვლავ რჩება ელემენტის ენერგომოხმარების დაზოგვა. ადრე ამ პრობლემის მოგვარებას ცდილობდნენ სიმლავრის რეგულირებით ან ელემენტის გაუმჯობესებით. ჩვენ მიერ შემოთავაზებული ქსელის მოდელის გამოყენებით მივაღწიეთ ქსელში ჩართულ სენსორებზე ენერგომოხმარების შემცირება და შესაბამისად დამაბოლოებელი მოწყობილობების განტვირთვა. აგრეთვე, ქსელში გარკვეულ უბნებზე მივიღეთ დატვირთვის შემცირება. აღსანიშნავია, რომ ქსელის პრინციპის ფუნქციონირებიდან გამომდინარე შემცირდა ინფორმაციის დაყოვნებაც.

პრაქტიკული მნიშვნელობა:

1) წარდგენილი ალგორითმების ქსელში რეალიზების შემთხვევაში შესაძლებელია რადიოსისტემების არხების განტვირთვა და დაყოვნების შემცირება.

2) წარდგენილი სისტემის ინტეგრაცია არსებულ ქსელებთან შესაძლებელია ეტაპობრივად სხვა მოწყობილობებთან თავსებადობის გათვალისწინებით.

3) ჩამოყალიბებულია მონაცემების დამუშავების მეთოდი, რომელის საშუალებით შესაძლებელია სენსორების ელემენტების ექსპლუატაციის დროის გაზრდა.

სამუშაოს აპრობაცია: სადისერტაციო ნაშრომის თემატიკასთან დაკავშირებული საკითხები აპრობირებული იქნა: 1. „სტუდენტური IoT ჰაკათონი“ - პროექტის პრეზენტაცია და ჰაკათონში გამარჯვება (ეროვნული სამეცნიერო ბიბლიოთეკა და Internet Society – Georgia, 14 ოქტომბერი, 2018 წელი, თბილისი); 2. ბრიტანეთის საბჭოს „THE BIG IDEA CHALLENGE“ : უმაღლეს სასწავლებლებში სამეწარმეო უნარების განვითარების პროგრამა (საქართველოს ტექნიკური უნივერსიტეტის და კილის უნივერსიტეტის ერთობლივი პროექტი № EV16048P8P - №2181, მაისი. 2021 წელი, თბილისი). 3. I, II და III კოლოქვიუმები და წინასწარი დაცვა. გარდა ამისა დისერტაციის თემაზე გამოქვეყნებულია 3 სტატია.

თავი I. ინფორმაციაზე ორიენტირებული ქსელის მიმოხილვა და მისი სახეობები. გამოყენების ეფექტურობა და რეალიზაცია

1.1 ინფორმაციაზე ორიენტირებული ქსელის ზოგადი მიმოხილვა

ბაზარზე შემოდის მრავალი IoT ტიპის უსადენო სენსორები. ეს სენსორები მუშაობენ უსადენო სისტემების ქსელებზე. ასეთებია: Wi-Fi, LoRaWan, Bluetooth, მობილური კავშირის რადიოსისტემები და ა.შ. სენსორებს, რომლებიც არიან განლაგებული რადიოშეღწევის წერტილებიდან მოშორებით, უწევთ რადიოსიგნალის გადაცემისთვის დიდი ენერგიის დახარჯვა. დახარჯული ენერგია შეიძლება მნიშვნელოვნად მეტი იყოს ვიდრე სენსორის მიკროკონტროლერის ენერგომოხმარება. ეს დამოკიდებულია იმაზეც, თუ რა პერიოდებით შემოდის მოთხოვნა სენსორებზე. თუ სენსორი პერიოდულად შეწუხებულია იგივე მონაცემების მრავალი მოთხოვნით, მისი ბატარეის ექსპლუატაციის ვადები შემცირებულია. ცხრ. 1-ზე იხილეთ სხვადასხვა რადიოსისტემის გადაცემის სიმძლავრე. გარდა ამისა, განმეორებითი მოთხოვნებით, რადიოარხის რესურსი არის ზედმეტად დატვირთული და არაეფექტური. ინფორმაციაზე ორიენტირებული ქსელების რადიოტექნოლოგიებთან შეთავსებით, ჩვენ შევძლებთ ასეთი პრობლემების გადაჭრას. განმორებითი მოთხოვნების დაკმაყოფილება გადავა სხვა საკაბელო მოწყობილობებზე. სენსორები და რადიოარხები არ იქნებიან ზედმეტად დატვირთულნი. მომხმარებლის მონაცემების მიღების დაყოვნებაც შემცირდება, ეს განსაკუთრებით სასარგებლოა, რადიოსისტემის დატვირთულ საათებში. უსაფრთხოების კუთხითაც კარგია ის, რომ სენსორის ბატარეის ექსპლუატირება ჰაკერის მიერ ვერ მოხერხდება.

ცხრ. 1. სიმძლავრის მოხმარება ტექნოლოგიების მიხედვით	
რადიოსისტემა	მაქსიმალური გადაცემის სიმძლავრე
GSM900	2ვ
GSM1800	1ვ
UMTS2100	125მვ
Wi-Fi	1ვ
Bluetooth	100მვ
LoRaWAN	25მვ
LTE	200მვ

ITU-T სექტორი მოიხსენიებს ICN-ს, FG IMT-2020 პროექტში, როგორც მომავლის ინტერნეტის ტექნოლოგიას. მეცნიერების ვარაუდით ICN-ს შეუძლია დააკმაყოფილოს ქსელის ყველა ის მოთხოვნა, რომელიც განხილულია IMT-2020-ის პროექტში. ICN ქსელი პრინციპულად განსხვავდება IPv4 ქსელისგან. აპრიორი, IPv4-გან განსხვავებით ICN-ის ქსელი ორიენტირებულია მხოლოდ კონტენტზე. კონტეტის დასახელების მაგალითი:

/movie_service/cisferi_mtebi/h264/786kbps/32kbps/Georgian

ფაქტობრივად, ICN-მა ამოიღო IP-ის ცნება და ფუნქციონირებს ე.წ. სახელების შრეზე. სადღეისოდ, ICN ბაზაზე მომუშავე ქსელებიდან ყველაზე დიდ ინტერესს იწვევს NDN-ის ქსელი. მოგეხსენებათ, რომ ICN-ის ბაზაზე მომუშავე ქსელები ერთმანეთისგან განსხვავდებიან მხოლოდ გამოყენებული პროტოკოლებით. ICN ტიპის ქსელი პირობითად შედგება: კონტენტის მწარმოებლისგან, კონტენტის მომხმარებლებისგან და აგრეთვე შუავამალი კვანძებისგან. [1] ცხრ. 2-ზე ნაჩვენებია NDN-ის ძირითადი უპირატესობები და ნაკლებობები IPv4-თან შედარებით.

ცხრ. 2. NDN-ის ძირითადი უპირატესობები და ნაკლებობები IPv4-თან შედარებით		
ტექნოლოგია	NDN	IPv4
ქეშირების საშუალება	დიახ	არა
გამოთვლითი რესურსი	საშუალო	მცირე
ქსელის კვანძების განტვირთვა	დიახ	არა
მყარი დისკის უტილიზაცია	მაღალი	მცირე
დაყოვნება	განმეორებით მოთხოვნების უზრუნველყოფა გაცილებით ნაკლები დაყოვნებით.	ერთჯერად მოთხოვნებზე მცირედ ნაკლები დაყოვნება
უსაფრთხოება	ინტეგრირებულია პკაეტების ხელმოწერების პრინციპზე	სჭირდება დამატებითი პროტოკოლები

21-ე საუკუნის პირველი მეოთხედის ბოლოში ინფორმაციული ტრაფიკის მოთხოვნა გაიზარდა და შესაბამისად გახდა საჭირო ახალი სიხშირეების გამოყოფა, სიგნალების დამუშავების ახალი მეთოდების შექმნა რათა გაზრდილიყო ინფორმაციის გადაცემის სიჩქარე. მილიმეტრული დიაპაზონის სიხშირეები ფართოდ იქნება გამოყენებული მე-5 თაობის მობილურ სისტემებში, რაც ხელს შეუწყობს ჰეტეროგენული ქსელების განვითარებას. რადიომაგნიტური ტალღის ბუნებიდან გამომდინარე ზემოთ აღნიშნულ დიაპაზონში მომუშავე რადიოსიხშირეებს აქვთ დიდი მილევა და ამიტომ ისინი მუშაობენ მოკლე მანძილზე და მხოლოდ პირდაპირი ხედვის არეში.[2] მილიმეტრული დიაპაზონი ძალიან მგრძნობიარეა ამინდის მიმართ. წვიმა, თოვლი გარკვეულ ზეგავლენას ახდენს სერვისზე. მოგეხსენებათ, რომ დიდი მოთხოვნაა ინტერნეტის სიჩქარეზე. უამინდობამ კი შესაძლოა გაუარესოს მომხმარებლის რადიო პირობები და შესაბამისად

იმუშავებს დაბალი რიგის მოდულაცია რაც გამოიწვევს სერვისის გაურესებას. იმის გამო, რომ ახალი სპექტრის გამოყოფა საგრძნობლად ვერ ზრდის სისტემის ტევადობას საჭიროა ახალი მეთოდების დანერგვა. ასეთია ICN-ის ინტეგრაცია LTE-ის ან შემდგომი თაობის მობილურ სისტემებთან. LTE-ის EPC-ის კვანძებისთვის უნდა განისაზღვროს შესაბამისი ICN სტანდარტები და მექანიზმები. მონაცემების დაქეშირების ხარჯზე შემცირდება დატვირთვა რადიო ინტერფეისებზე. მობილური ტელეფონებისა და სენსორების დიდი ნაწილი არის მონაცემზე ორიენტირებული. ICN-ის საშუალებით, ბუნებრივად, შესაძლო იქნება რადიო არხებზე განმეორებითი მონაცემების გაგზავნის შემცირება. გარდა განმეორებითი მოთხოვნების, ასევე შესაძლებელი გახდება სასიგნალო ინფორმაციის განმეორებითი გადაცემის ე.წ. რეტრანზმისიების შემცირება EPC-ის ქსელის სხვადასხვა კვანძზე.

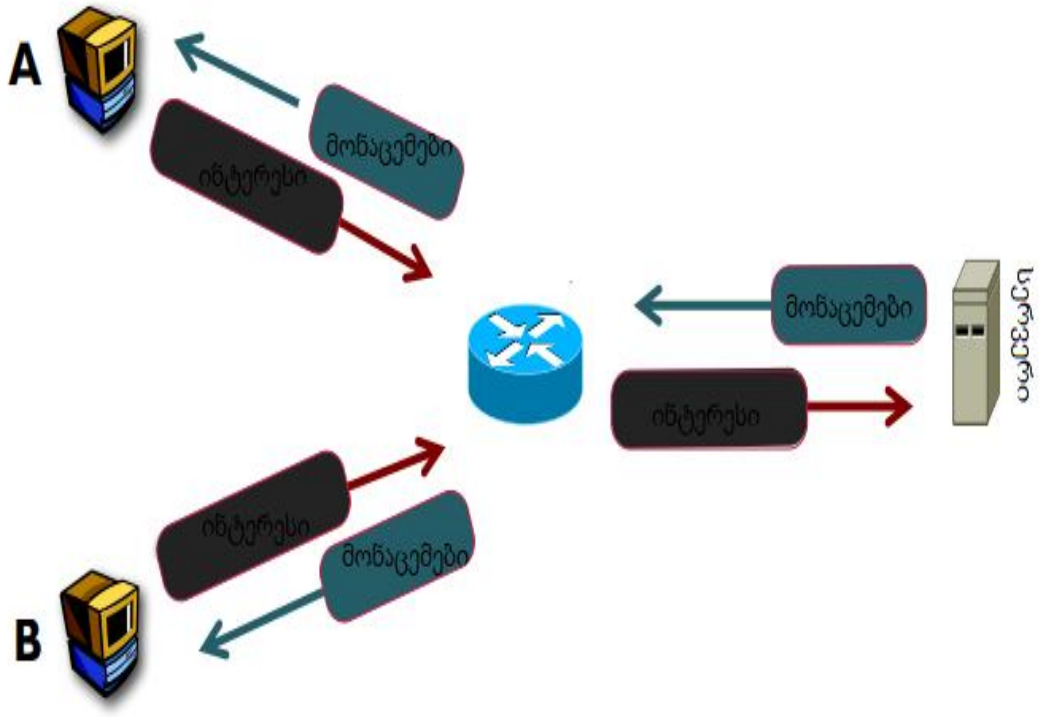
1.2 NDN ქსელის მოკლე მიმოხილვა

NDN-ი ცნობილია, როგორც მომავლის ინტერნეტი, თავის მხრივ ეს ტერმინი დაკავშირებულია ინფორმაციაზე ორიენტირებულ ქსელთან ე.წ. “Information - Centric Network”-თან, რომელიც იყო შემოთავაზებული ვან ჯაკობსონის მიერ, ტერმინი პირველად 2006 წელს გამოჩნდა და შემდგომ საფუძველი ჩაუყარა NDN-ის განვითარებას. ემპერიული კვლევები იყო ჩატარებული რათა აღმოეფხვრათ რიგი პრობლემები, რომლებიც მკვეთრად გაიზარდა ინტერნეტის განვითარების შედეგად. NDN-ის მოტივაცია დაფუძნებულია იმ ფაქტზე, რომ არსებული გლობალური ინტერნეტით სარგებლობის უმეტესი ნაწილი, ინფორმაციული ვიდეოტრაფიკის თითქმის 75%-ს შეადგენს და გადაიცემა წერტილი-მრავალწერტილი კონფიგურაციით (Point-to-Multipoint). სწორედ, ამიტომ NDN-ის ქსელის რეალიზაციას გააჩნია რიგი უპირატესობები ე.წ. კონტენტის ქეშირების ხარჯზე, რაც გულიხმობს ქსელის გამანაწილებელ მოწყობილობებზე

საჭიროების შემთხვევაში, ქსელიდან მოპოვებული გარკვეული ინფრომაციის შენახვას გარკვეული დროით და ამ ინფრომაციის მიწოდებას საჭიროების შემთხვევაში იმ მომხმარებლებთან, რომლებიც იმყოფებიან ქსელის გამანაწილებელი მოწყობილობის ზონაში.

სახელდარქმეული მონაცემთა ინტერნეტი მოიაზრებს ინტერნეტის დამისამართების პრინციპის შეცვლას, კერძოდ ციფრული დამისამართების ნაცვლად სახელების გამოყენებას. იმის ნაცვლად რომ მომხმარებელი დაუკავშირდეს კონკრეტულ ფიზიკურ მოწყობილობას, ამ შემთხევაში ის იქნება მხოლოდ ორიენტირებული კონტენტზე. მომხმარებლისთვის არ იქნება მნიშვნელოვანი საიდან მიიღებს ინფორმაციას. NDN-ს საშუალებით, ქსელში კონტენტის მოთხოვნის შედეგად ყველა მოწყობილობას ექნება ე.წ. ინტერესი. ნაგულიხმევია, რომ თითოეული მოწყობილობა აწარმოებს ჩანაწერს იმ კონტენტზე, რომელზეც იყო შემოსული რაიმე სახის მოთხოვნა. ინფორმაციული მოთხოვნის მიღების შემდეგ მოწყობილობები თვითონვე ინტერესდებიან შესაბამისი კონტენტით და ითხოვენ მას სხვა მომიჯნავე მოწყობილობებისგან.[3] ქსელში მოპოვებული კონტენტის მიღების შემდეგ მოწყობილობები დროებით ინახავენ მას თავიანთ ე,წ. „cache-ში“ და განმეორებითი მოთხოვნის მიღების შემთხევაში გააზიარებენ ამ კონტენტს. NDN-ი დაცულია ასიმეტრული კრიპტოგრაფიული გზით. კონტენტი შემოწმდება სერტიფიკატის საშუალებით. ნახ.1-ზე ნაჩვენებია NDN ქსელის ზოგადი სტრუქტურა.

NDN-ის ქსელის სემანტიკა არის რადიკალურად განსხვავებული TCP/IP პროტოკოლისგან და მოიაზრებს ინტერნეტის დამისამართების პრინციპის შეცვლას, კერძოდ ციფრული დამისამართების ნაცვლად სახელების გამოყენებას. ეს იმაზე მეტყველებს რომ მომხმარებელი არ არის ორიენტირებული კონკრეტულ ფიზიკურ მოწყობილობის მისამართზე და მის ადგილმდებარეობაზე, ის იქნება მხოლოდ ორიენტირებული „კონტენტზე“.



ნახ. 1. NDN ქსელი

ქვემოთ განხილულია NDN-ის უპირატესობები:

1)NDN-ის ბაზაზე შემუშავებული ალგორითმის საფუძველზე შესაძლებელია დაყოვნების შემცირება.

2)NDN-ის ქსელი წარმოადგენს ჰეტეროგენულ ქსელს, აქედან გამომდინარე საშუალებას გვაძლევს გამოვიყენოთ ნივთების ინტერნეტში სხვადასხვა ტიპის ტექნოლოგიებთან ერთად.

3) NDN გამოიყენება ნივთების ინტერნეტში და გარკეულწილად, LTE-ში არსებული “Small Cell“-ის როლს თამაშობს. მას შეუძლია მოგვცეს ახალი დამატებითი ტრაფიკის შემოღება, მაგრამ ეს ტრაფიკი იქნება ლოკალური და არ შეუშლის ხელს ქსელში იმ მონაკვეთებზე სადაც ტრაფიკის კონცენტრაცია არის მაღალი და ამავე დროულად გარკვეულწილად გაათანაბრებს ქსელში ტრაფიკის განაწილებას. იგი გაზრდის ქსელის გამტარუნარიანობას. ამიტომ ჩემი აზრით ასეთი ტიპის მიდგომები უნდა დაინერგოს მასობრივად.

4) NDN-ის TCP/IP სისტემასთან ერთობლივი გამოყენება საშუალებას მოგვცემს გავზარდოთ ქსელის გამტარუნარიანობა. ამიტომ უმჯობესია გამოვიყენოთ ეს ორი პროტოკოლი ერთობლივად.

1.2.1 როუტერის არქიტექტურა NDN-ის ქსელში

იმისთვის რომ აბონენტის მიერ გაგზავნილი ე.წ. ინტერესი დავაკმაყოფილოთ, თავდაპირველად NDN-ის ქსელში მოთხვონა ინტერესის სახით იგზავნება ლოკალურ როუტერზე, თითოეულ როუტერს გააჩნია მონაცემთა 3 სახეობის სტრუქტურა:

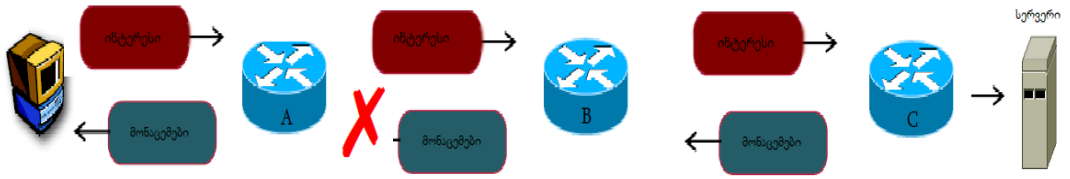
Pending Interest Table (PIT) მოლოდინის ინტერესთა ცხრილი: ინახება ყველა ის ინტერესი, რომელიც ქსელში როუტერის მიერ იყო გაგზავნილი, მაგრამ ჯერ არ დაკმაყოფილდა. აგრეთვე, თითოეულ ჩანაწერს თან ახლავს როუტერის შემავალი/გამავალი ინტერფეისი.

Forwarding Information Base (FIB): ინფორმაციის გადაცემის ბაზა, TCP/IP-ის მარშრუტიზაციის ცხრილის მსგავსად მუშაობს. FIB წარმოადგენს მარშრუტიზაციის ცხრილს, რომელიც აკავშირებს ინტერესთა პაკეტს შესაბამის მონაცემთა პაკეტთან და ინტერფეისთან, რის საშუალებითაც ის უნდა გაიგზავნოს ქსელში.

Content Store (CS) - კონტენტის საცავი: მონაცემთა პაკეტების ე.წ. მარგი ტვირთის დროებითი შემნახველი ინფორმაციის ბაზა.[1]

იმ შემთხვევაში, თუკი ქსელში მომხმარებლის მიერ მოთხოვნილი ინფორმაციის მოძიება ვერ მოხერხდა და ეს სრულიად შესაძლებელია, მაგალითად როდესაც რომელიმე კვანძიდან გადაცემული ინფორმაცია აბონენტამდე ვერ აღწევს, იკარგება გზაში, ლინკის მოულოდნელი დაკარგვის შედეგად. ცხადია, ქსელის მომხმარებელმა კიდევ ერთხელ უნდა გააგზავნოს მოთხოვნა ქსელში, რათა მიიღოს ინფორმაცია. სამაგიეროდ, NDN ქსელის მომხმარებელი სასურველი ინფორმაცის

მოპოვებას შეძლებს პირველივე კვანძიდან რომელზედაც იყო დაქეშირებული ინფორმაცია ლინკის დაკარგვამდე. ნახ.2-ზე იხილეთ მაგალითი, სადაც სერვერიდან წამოსული მონაცემთა პაკეტი დაკარგვამდე, იყო დაქეშირებული B პუნქტში ამიტომ, ხელახალი მოთხოვნის შედეგად აბონენტი სასურველ ინფორმაციას წამოიღებს B პუნქტიდან.



ნახ. 2. სერვერიდან წამოსული მონაცემთა პაკეტის მაგალითი

აღსანიშნავია, რომ ქსელის რომელიმე მონაკვეთზე დიდი გადატვირთვის გამო შესაძლოა აბონენტის მიერ გაგზავნილი მოთხოვნა დაიკარგოს ხელოვნურად, ეს მეთოდი არის ქსელის ერთ-ერთი თავდაცვის მექანიზმი, რომელიც დაგვიცვას ჰაკერული “DoS” შეტევებისგან.

1.2.2 NDN ქსელის მუშაობის პრინციპი და უპირატესობები

როდესაც NDN-ის ქსელში, ლოკალურ როუტერზე შემოდის ინტერესთა პაკეტი, თავდაპირველად NDN-ის როუტერი ამოწმებს CS-ს ცხრილს. სასურველი ინფორმაციის ცხრილში მოძებნის შემთხვევაში, ამოიღებს მოთხოვნილი ინტერესის შესაბამის მონაცემებს და გაუშვებს ქსელში ინტერესის შესაბამისი ინტერფეისით. ამას აკეთებს FIB-ის ბაზაში არსებული ცხრილის თანახმად. წინააღმდეგ შემთხვევაში, თუკი ჩანაწერი არ მოიძიება მაშინ მოთხოვნა გაიგზავნება როუტერის ადაპტური გადაცემის ალგორითმის შესაბამისად. აღსანიშნავია, ის რომ სხვადასხვა კვანძებიდან NDN-ის როუტერზე, ერთსა და იმავე დროის ინტერვალში, ერთი და იგივე მოთხოვნა რომ შემოვიდეს, როუტერი მოემსახურება

მხოლოდ პირველი ინტერფეისიდან მოსულ მოთხოვნას. ალბათობა იმისა რომ დროის ერთსა და იმავე ინტერვალში როუტერმა მიიღოს ერთზე მეტი ერთნაირი მოთხოვნა არის ძალიან მცირე. თეორიულად შეიძლება ისეთი ალგორითმის დაწერა, რომლის საფუძველზე შესაძლებელი გახდება რამდენიმე აბონენტის ერთდროული მომხასურება, აბსოლუტურად ერთნაირი მოთხვონის არსებობის შემთხვევაშიც კი. ცხადია, გარკვეულ ხარჯებთან იქნება დაკავშირებული.

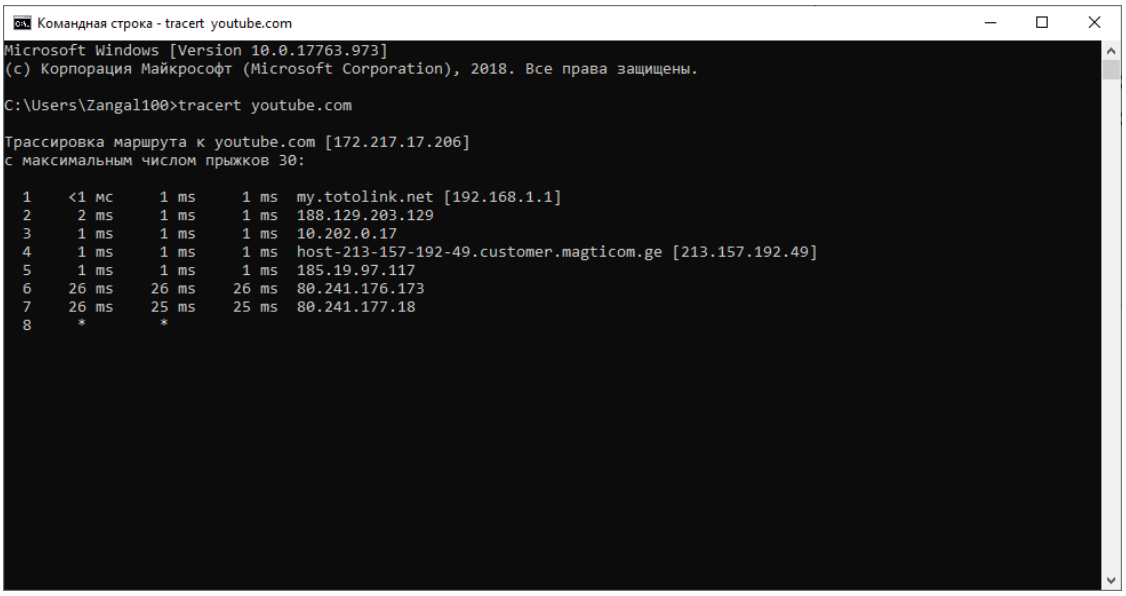
განვიხილოთ შემდეგი მაგალითი: დავუშვათ გვაქვს ქსელი და პირობითად A პუნქტში მყოფ აბონენტს სურს ამინდის პროგნოზის გაგება, შესაბამისად ქსელში იგზავნება მოთხოვნა ე.წ. ინტერესი. ჩვენს მიერ კონკრეტულ განხილულ მაგალითში მოთხოვნა პირველ რიგში გაიგზავნება ლოკალურ როუტერზე. ის თავდაპირველად გადაამოწმებს Content Store-ში არსებულ ბაზას. მონაცემთა ბაზაში შესაბამისი მოთხვოვნის სასურველი ინფორმაციის მოძიების შემთხვევაში ლოკალური როუტერი აბონენტს დაუბრუნებს პასუხს იმავე ინტერფეისით და შესაბამისად მოხდება აბონენტის სერვისით დაკმაყოფილება. იმ შემთხევაში თუკი როუტერის ცხრილში არ მოიძებნება ე.წ. ინტერესის შესაბამისი ინფორმაცია, მაშინ როუტერი გადასცემს ამ მოთხოვნას შემდეგ კვანძზე. ზემოთაღნიშნული ოპერაცია იქამდე გაგრძელდება სანამ რომელიმე მოწყობილობა არ გამოეხმაურება და არ დაუბრუნებს შესაბამის მოთხოვნაზე პასუხს. აღსანიშნავია, ის ფაქტი რომ ქსელის როუტერიდან წამოღებული ინფრომაცია შეინახება ყველა იმ მოწყობილობაზე, რომლებზეც გავლა მას მოუწია აბონენტამდე ინფორმაციის გადაცემის პროცესში. ამავე აბონენტის მეორედ მოთხოვნის გაგზავნის შემდეგ, კერძოდ კი ამინდის პროგნოზის გაგებისთვის, ინტერესის პაკეტი NDN-ის ქსელში სერვერამდე აღარ გაიგზავნება, არამედ მას მოემსახურება ლოკალური როუტერი.

ქვემოთ ჩამოთვლილია NDN-ის უპირატესობები:

1) შემცირდება დაყოვნების დრო ვინაიდან აბონენტის მიერ გაგზავნილი მოთხოვნა, აღარ იგზავნება სერვერზე. აბონენტს ემსახურება ლოკალური როუტერი. მარტივი მისახვედრია, რომ ლოკალური როუტერის შემთხვევაში ქსელში გავრცელებული ინფორმაციის დრო შემცირდება, რადგან ის ნაკლებ კვანძს გაივლის. სწორედ ამიტომ შემცირდება მონაცემების მიღების დაყოვნება. იხილეთ პაკეტის გამავალ გზაზე არსებული მოწყობილობების დაყოვნებები ნახ.3 -ზე. როგორც უკვე ავღნიშნეთ NDN ქსელის აბონენტი სასურველი ინფორმაციის ქსელში გარკვეული მიზეზების დაკარგვის შედეგად, შესაძლოა მოიპოვოს მოთხოვნილი ინფორმაცია ყველაზე ახლო მდებარე როუტერიდან ე.წ. NDN ქსელის ქეშირების ფუნქციიდან გამომდინარე.
2) სიჩქარე გაიზრდება ქსელის გარკვეულ სეგმენტზე, იქიდან გამომდინარე რომ ჩვენ NDN-ის გამოყენებით შევძლებთ ტრაფიკის მოხსნას ლოკალურად, რის შედეგადაც გლობალურ ქსელში გარკვეულ უბნებზე გადატვირთვებისგან თავს ავიცილებთ, და აგრეთვე გლობალურად გაიზრდება ქსელის გამტარუნარიანობა ანუ ტევადობა, ბევრი ლოკალური ტრაფიკის შემოღებით.
3) არ გვჭირდება IPv6-ზე გადასვლა იქიდან გამომდინარე რომ იცვლება ინტერნეტის სემანტიკა, რაც გულისხმობს ინფორმაციის მოძიებას ქსელში ე.წ. ინტერესებით და არა რაიმე კონკრეტულ მისამართზე მოთხვონის გაგზავნის შედეგად.
4) NDN-ში სახელების სივრცე არ არის შეზღუდული. ინტერესთა პაკეტის მაქსიმალური ზომა შეადგენს 8 კბტ-ს. NDN-ში გამოიყენება “საზოგადო IP” მისამართები[4]

- A 10.0.0.0 — 10.255.255.255; 255.0.0.0
- B 172.16.0.0 — 172.31.255.255; 255.255.0.0
- C 192.168.0.0 — 192.168.255.255; 255.255.255.0

ზოგიერთ ადამიანს შეიძლება არ მოეწონოს IPv6-ზე არ გადასვლის ტენდენცია, შესაბამისად ჩვენ შეგვიძლია გამოვიყენოთ ჰეტეროგენული ქსელი, ანუ მოვახდინოთ NDN-ის და არსებული ინტერნეტის შერწყმა და მაგალითად შევქმნათ ისეთი პროტოკოლი, რომელიც გარკვეული ალგორითმის დამუშავების შედეგად მიიღებს გადაწყვეტილებას თუ რომელი ქსელი გამოიყენოს ამა თუ იმ სიტუაციაში, რათა გავზარდოთ მაქსიმალურად ქსელის გამოყენების ეფექტურობა, და ხარისხიანად მოვემსახუროთ აბონენტებს.



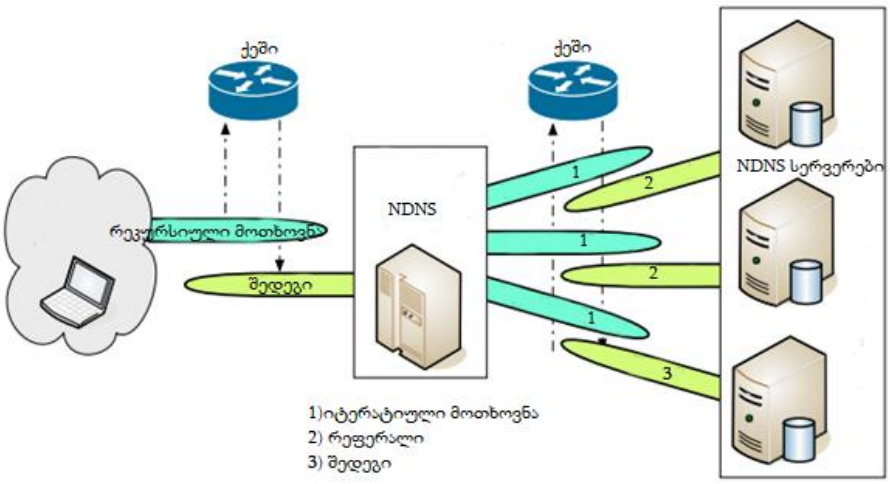
ნახ. 3. გაშვებულია Tracert ბრძანება

აქვე მინდა ავღნიშნო წარმოქმნილი უარყოფითი მომენტები, რომელიც თან ახლავს NDN-ს: იქიდან გამომდინარე რომ დიდი მოცულობის ინფორმაციის შენახვა შეიძლება მოგვიწიოს როუტერებზე. მოწყობილობებზე უნდა გაიზარდოს მყარი დისკის მოცულობა, რამაც შესაძლებელია შეამციროს ინფორმაციის დამუშავების სიჩქარე რაც გამოიწვეს დაყოვნების გაზრდას, ამიტომ სასურველია SSD-ების გამოყენება, მაგრამ ფინანსებთან არის დაკავშირებული და შეიძლება იყოს არა

მომგებიანი. ამის გამო ვფიქრობ, რომ საუკეთესო შედეგს მოგვცემს ღრუბლოვანი ტექნოლოგია.

1.3 NDNS & ქსელის უსაფრთხოება

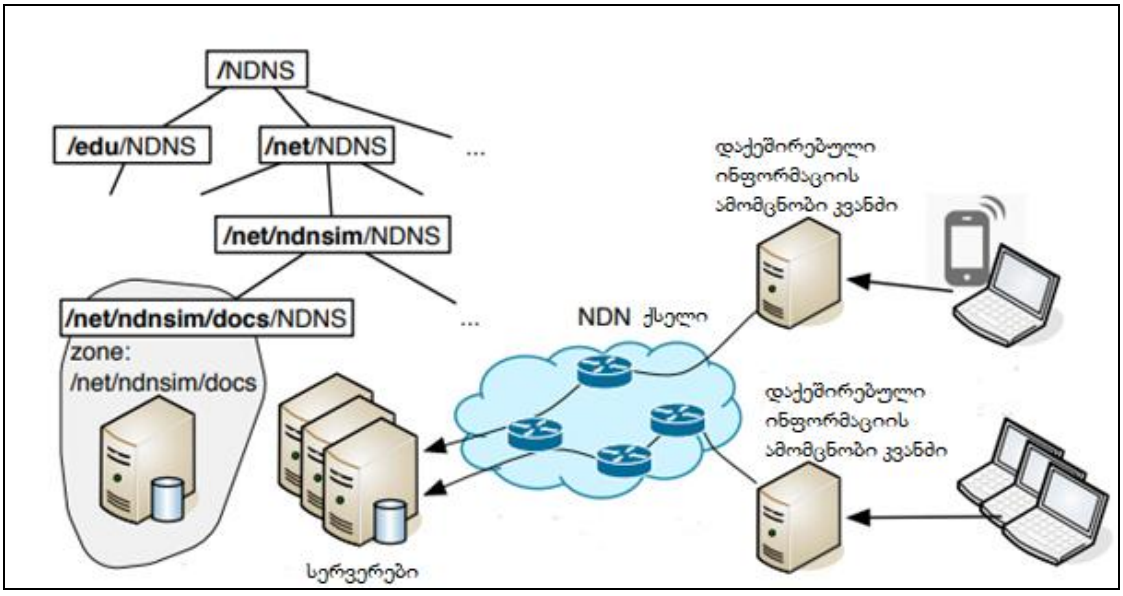
TCP/IP პროტოკოლის თანახმად, DNS-მუშაობს OSI-ს მოდელის სააპლიკაციო შრეზე. NDN-ის ქსელში DNS-ის როლს ასრულებს NDNS, რომელიც DNS-გან განსხვავებით მუშაობს ქსელურ შრეზე.[5] იხილეთ ნახ.4-ზე NDNS-ის ზოგადი არქიტექტურა.



ნახ. 4. ნაჩვენებია NDNS-ის ზოგადი არქიტექტურა

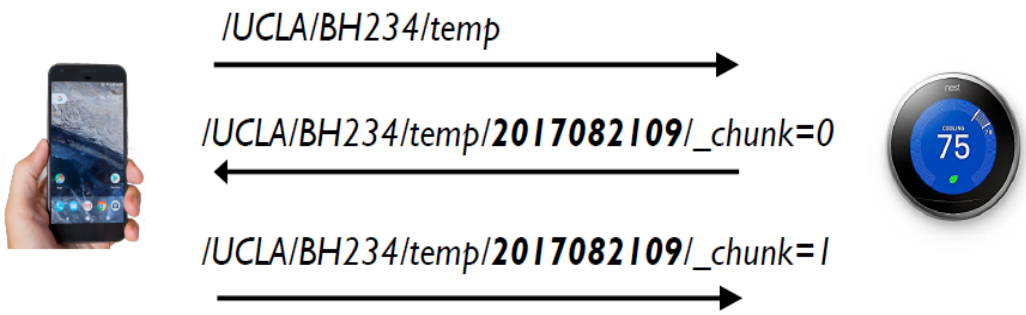
NDN-ის ქსელში აბონენტის მიერ აპლიკაციის საშუალებით დაგენერირებული ინფორმაციული მოთხვონა, საწყის ეტაპზე „Stub Resolver“-ით რეკურსიული მოთხოვნის ფორმატით „Caching NDNS Resolver“ - ის მიმართულებით გადაიცემა, მაგრამ შესაძლოა NDNS-მდე მოთხოვნა არ გავრცელდეს, და მოიძიოს რომელიმე როუტერზე. თუკი მომხმარებლის მიერ გაგზავნილი მოთხოვნა არ იყო დაქეშირებული არც ერთ როუტერზე, ამ შემთხვევაში „Caching NDNS Resolver“-ი ეცდება რომ მოძებნოს ინფორმაცია თავის ცხრილში, ხოლო ინფორმაციის არ არსებობის

შემთხვევაში იტერატიული მოთხოვნის ბრძანებით გააგზავნის “Authoritative NDNS Servers”-კენ. იმისათვის, რომ ინფორმაციული მოთხოვნა ანუ ინტერესი მივიდეს ადრესატამდე, მომხმარებლის მიერ გაგზავნილ პაკეტში უნდა ჩაიწეროს ე.წ. ზონის სახელი, ანუ დომენის სახელი მსგავსად TCP/IP-ში. სწორედ ამიტომ NDNS Caching Resolve-მა წინასწარ უნდა იცოდეს ეს ზონები, რათა მისცეს მომხმარებლის მიერ გამოგზავნილ ინტერესთა პაკეტს სწორი მიმართულება NDN-ის ქსელში.[5] იხილეთ NDNS ელემენტები ნახ.5-ზე



ნახ. 5. ნაჩვენებია NDNS ელემენტები

„Stub resolver“-ი წარმოადგენს NDNS-ის კომპონენტს, რომელიც ასრულებს შუამავალ როლს სააპლიკაციო პროგრამებსა და „NDNS Resolver“-ს შორის. „Stub resolver“-ი TCP/IP პროტოკოლის შემთხვევაში გარდაქმნის „domain name“-ს IP მისამართებად. NDNS-ის მოთხოვნის პაკეტის ფორმატი იხილეთ ნახ. 6-ზე.



ნახ. 6. NDNS-ის მოთხოვნის პაკეტის ფორმატი[6]

NDNS-ის მუშაობის ორი რეჟიმი არსებობს:

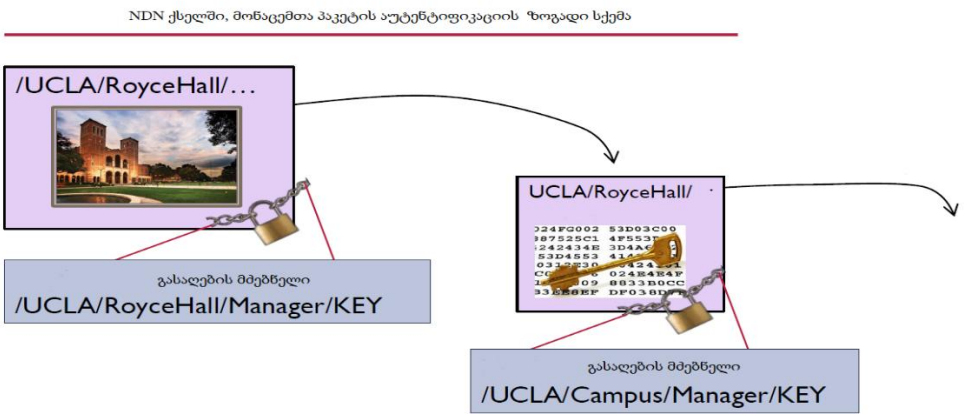
იტერატიული - თუკი სერვერი პასუხისმგებელია რომელე დომენის ზონაზე ის თავად აბრუნებს პასუხს. წინააღმდეგ შემთხვევაში აბრუნებს DNS-ის სერვერის მისამართს, რომელსაც თან ახლავს დაწვრილებითი ინფორმაცია.

რეკურსიული - სერვერი თვითონ ასრულებს მოთხოვნებს სხვა DNS სერვეერბთან, რათა მოძებნოს სწორედ ის რომელიც მას სჭირდება.

NDN-ის ქსელში აბონენტის მიერ გაგზავნილ მოთხოვნის პაკეტში აუცილებლად ჩაიწერება სასურველი ინფორმაციის სახელი ან პრეფიქსი, რათა მოხდეს სასურველი ინფორმაციის იდენტიფიკაცია, რომელიც ხორციელდება ქსელურ შრეზე. NDN-ის ქსელში თითოეული როუტერის ერთ-ერთი განუყოფელი ლოგიკური ნაწილია FIB ცხრილი, ის აუცილებლად მოიცავს საასურველი ინფორმაციის ფრეფიქსებს, ინტერფეისების სიას.

NDN-ში ინფორმაციის მწარმოებლის მიერ იქმნება ე.წ. ციფრული ხელმოწერა, რომელიც უნდა დაისვას შესაბამის მონაცემთა პაკეტზე ფორმირებისთანავე. კრიპტოგრაფიული ხელმოწერა აერთიანებს მონაცემთა დასახელებას შესაბამის კონტენტთან. ციფრული ხელმოწერის ველში ჩაიწერება გასაღების მძებნელი - “Key locator”-ი, რომელიც გამოიყენება ხელმოწერის ვერიფიკაციისთვის მიმღებ მხარეზე. გასაღები წარმოადგენს

ჩვეულებრივ მონაცემთა პაკეტს, რომელიც მოიცავს ე.წ. “Public Key bits” საჯარო გასაღების ბიტებს შესაბამის ხელმოწერასთან ერთად, რომელიც თავის მხრივს მოიცავს შესაბამის “Key locator”-ს. ზემოთაღნიშნული “Key locator”-ების ჯაჭვი წარმოადგენს უსაფრთხოების სერტიფიკატს, რომლის წასაკითხად ქსელის მომხმარებელს უნდა ქონდეს ე.წ. “anchor key” ღუზა გასაღების საშუალებით გახსნის პაკეტს და ამოიღებს მარგი ტვირთის ინფორმაციას. ამ ინფორმაციის ყველა სხვა როუტერზე შენახვის შემთხვევაში, მონაცემთა პაკეტი იქნება დაშიფრული და სხვა აბონენტები ვერ მოახერხებენ ამ ინფორმაციის წაკითხვას სპეციალური ციფრული გასაღების გარეშე. ნახ. 7-ზე იხილეთ NDN ქსელში მოანცემთა პაკეტის აუტენტიფიკაციის ზოგადი სქემა.



ნახ. 7. NDN-ის მოანცემთა პაკეტის აუტენტიფიკაციის ზოგადი სქემა[6]

1.4 IoT ქსელის მიმოხილვა

ნივთების ინტერნეტი მოიაზრებს ყველაფრის ყველაფერთან დაკავშირებას ინტერნეტის საშუალებით. IoT - „ნივთების ინტერნეტი“, როგორც ასეთი ცნება არის ძალიან მარტივი. წარმოვიდგინოთ, რომ ყველა ნივთს ჩვენს გარშემო აქვს მინიატურული საიდენტიფიკაციო და სენსორული მოწყობილობა. კავშირის არხების საშუალებით შესაძლებელია არა მარტო ამ ობიექტების და მათი მახასიათებლების მონიტორინგი

სივრცეში და დროში არამედ მათი მართვაც. ინფოკომუნიკაციის მიხედვით “ნივთების ინტერნეტის” ფორმულირება ჩაიწერება შემდეგნაირად: სენსორები, მონაცემები, ქსელები, მომსახურეობა.

ნივთების ინტერნეტის ქვეშ მოიაზრება მოწყობილობების ერთობლიობა, რომელიც გაერთიანებულია ქსელში, ხოლო ეს მოწყობილობები კი ერთამენეთთან დაკავშირებული არიან ნებისმიერი თავისუფალი კავშირის არხებით და იყენებენ ურთიერთქმედების სხვადასხვა პროტოკოლებს და მხოლოდ IP პროტოკოლს გლობალურ ქსელთან წვდომისთვის. IoT ტექნოლოგია პირველად გამოჩნდა 2008-2009 წლებში. IoT-ის მთავარი ამოცანაა მოწყობილობების ერთმანეთთან დაკავშირება. IoT ტექნოლოგია გულისხმობს “ობიექტების მოჭკვიანებას“. სადღეისოდ, ინტერნეტში ჩართული მოწყობილობების რაოდენობა ბევრად აღემატება მსოფლიოს მოსახლეობის რაოდენობას.

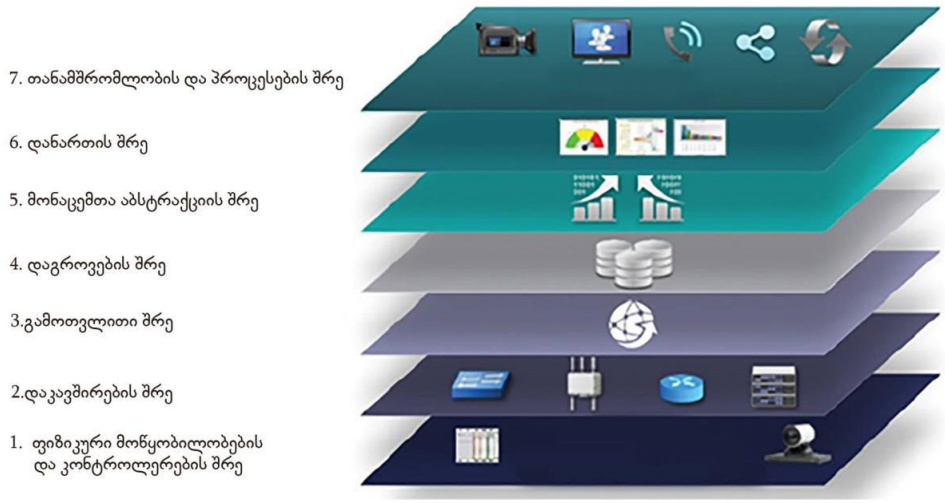
ლოგისტიკის სისტემის ოპტიმიზაციის აუცილებლობამ განაპირობა IoT-ის შექმნა რის შედეგადაც კლიენტებისთვის პროდუქციის მიწოდების პროცესის მართვა გახდა უფრო მარტივი და მოხერხებული.

ინოვაციათა მეორე ტალღა ხარჯების შემცირებით იყო განპირობებული ვიდეომეთვალყურეობის სისტემებში, უსაფრთხოებაში, და ა.შ.

ინოვაციათა მესამე ტალღა იყო გამოწვეული გეოლოკაციური სერვისების აუცილებლობით. ინოვაციათა მეოთხე ტალღა განპირობებულია პროცესების ავტომატიზაციით ანუ ადამიანის შრომის შემსუბუქებით .

მომავალში ტექნოლოგია უფრო დაიხვეწება და განვითარდება. უფრო მეტი მოწყობილობა იქნება ჩართული საერთო ქსელში, შეიქმნება მომავალის ქსელები „Future Networks” წრიული ტოპოლოგიით, რომლებშიც იქნება თავმოყრილი დიდი რაოდენობით სხვადასხვა ტიპის სენსორები, გაზომვის საშუალებები, მართვის მოწყობილობები და ა.შ.

2014 წ. IoT-ის საერთაშორისო ფორუმზე (Cisco, IBM, Rockwell automation) მიერ ჩატარებულ კონფერენციაზე გამოიქვეყნა IoT-ის არქიტექტურული ეტალონური მოდელი. ამ მოდელის თითოეული დონე ასრულებს სპეციფიკურ ფუნქციას.

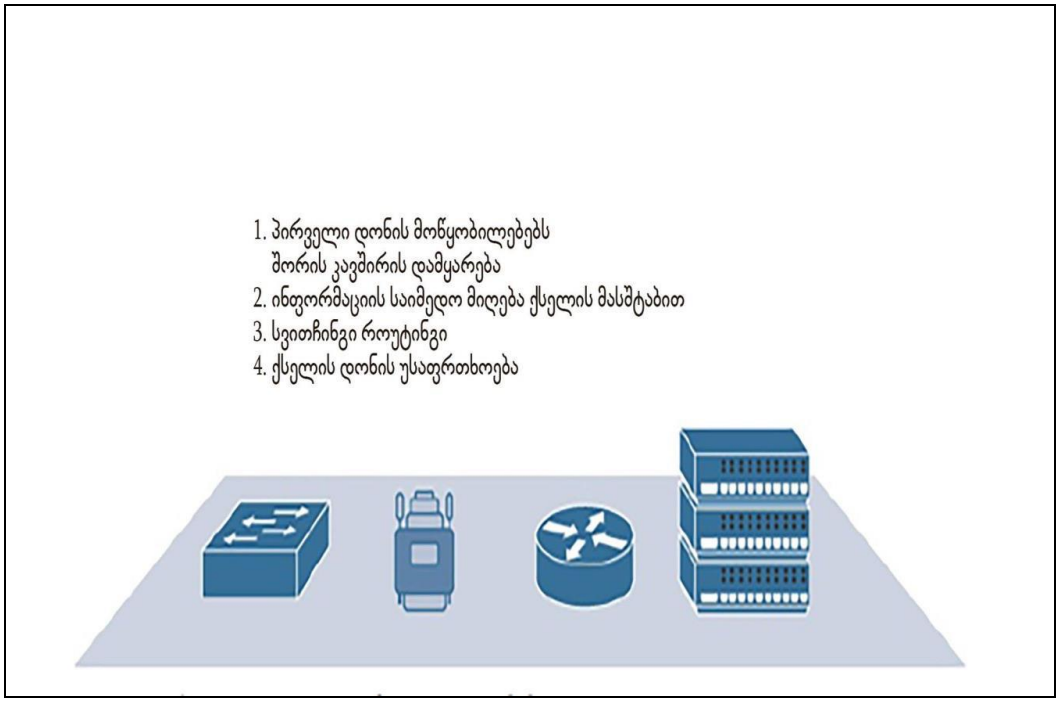


ნახ. 8. IoT-ის არქიტექტურული მოდელი

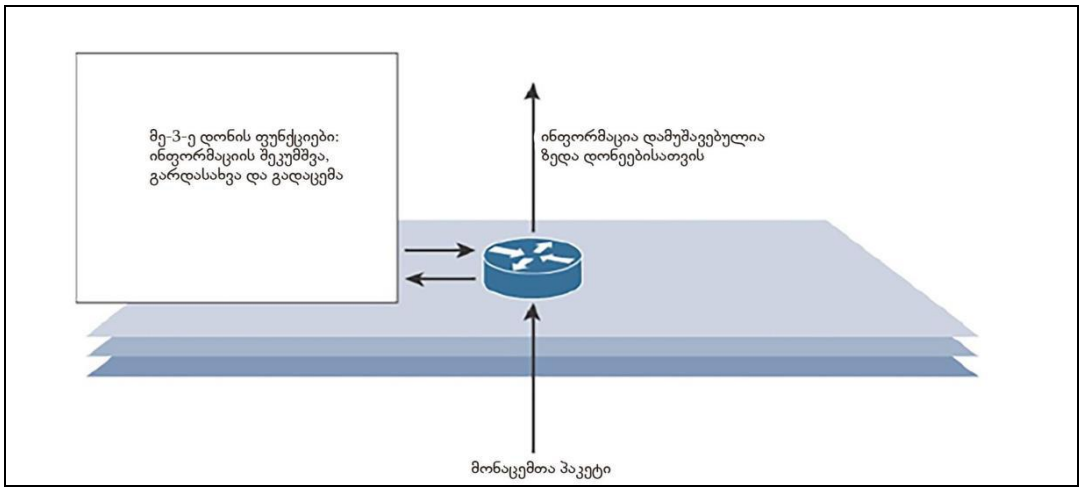
IoT-ის ეტალონური მოდელის პირველი დონე - ფიზიკური მოწყობილობების და კონტროლერების დონე. ეს დონე წარმოადგენს ამ ტექნოლოგიის ფუძეს. ის მოიცავს საბოლოო წერტილების მოწყობილობებს და სენსორებს, რომლებიც გადასცემენ და მიიღებენ ინფორმაციას. ამ ნივთების ზომები იცვლება მიკროსკოპური სენსორებიდან გიგანტურ მანქანებამდე. მათი მთავარი ფუნქციებია მონაცემთა წარმოქმნა და კონტროლი. ნახ. 8 - ზე ნაჩვენებია IoT-ის არქიტექტურული მოდელი.

დაკავშირების შრე - წარმოადგენს ეტალონური მოდელის მე-2 შრეს. ამ შრეზე დიდი ყურადღება ექცევა ყველა სახის დაკავშირებას. ამ დონის უმნიშვნელოვანესი ფუნქციებია: საიმედოობა და ინფორმაციის დროული გადაცემა. უფრო ფართო გაგებით რომ ვთქვათ, დაკავშირების დონე მოიცავს ინფორმაციის გადაცემას პირველი დონის მოწყობილობებს და

ქსელს შორის, და ასევე ქსელსა და ინფორმაციის დამუშავების ანუ მესამე დონეს შორის, აგრეთვე ეტალონური მოდელის მეორე შრე მოიცას საქსელო ყველა ელემენტს.



ნახ. 9. IoT-ის არქიტექტურული ეტალონური მოდელის მე-2-ე შრე და მისი ფუნქციები



ნახ. 10. ეტალონური მოდელის მე-3 შრე

გამოთვლითი შრე - წარმოადგენს ეტალონური მოდელის მე-3 დონეს, ამ დონეზე ხორციელდება: ინფორმაციის შეკუმშვა , ინფორმაციის გარდასახვა და შენახვა. ნახ.10-ზე ნაჩვენებია ეტალონური მოდელის მე-3 შრე.

დაგროვების შრე - ამ დონეზე ხორციელდება მესამე დონიდან მიიღებული მონაცემების შენახვა.

მონაცემთა აბსტრაქციის შრე - ამ დონეზე ხორციელდება ინფორმაციის გაერთიანება და შენახვა ერთ წყაროზე ან ინფორმაციის გაყოფა და შენახვა სხვადასხვა წყაროებზე, ვირტუალიზაციის საშუალებით.

დანართის შრე - მონაცემთა ინტერპრეტაცია პროგრამული აპლიკაციებით, აპლიკაციას შეუძლია მონიტორინგის განხორციელება.

თანამშრომლობის და პროცესების შრე - ამ დონეზე ხორციელდება ინფორმაციის მიტანა მომხმარებლამდე.

IoT-ის საფუძველზე შესაძლოა განხორციელებულ იქნას ჭკვიანი აპლიკაცია საქმიანობის ყველა სფეროში. განვიხილოთ რამდენიმე მაგალითი:

ჭკვიანი ქალაქი - ქალაქის ინფრასტრუქტურა და მუნიციპალური მომსახურეობები ასეთი როგორიც არის: განათლება, ჯანდაცვა, უსაფრთხოება, ეს სისტემები გახდება ერთმანეთთან დაკავშირებული და შესაბამისად მომსახურეობის ეფექტურობა გაიზრდება.

ჭკვიანი სახლი-სისტემა, რომელიც უზრუნველყოფს უსაფრთხოებას, კომფორტს და რესურსების ეკონომიურ განაწილებას.

ჭკვიანი ენერგეტიკა - იქნება უზუნველყოფილი ელექტრული ენერგიის წყაროდან მიმღებამდე ზუსტ დროსა და საჭირო რაოდენობით საიმედო და ხარისხიანი გადაცემა.

ჭკვიანი ტრანსპორტი-მგზავრების გადაადგილება სივრცის ერთი წერტილიდან მეორე წერტილში გახდება უფრო კომფორტული , სწრაფი და უსაფრთხო.

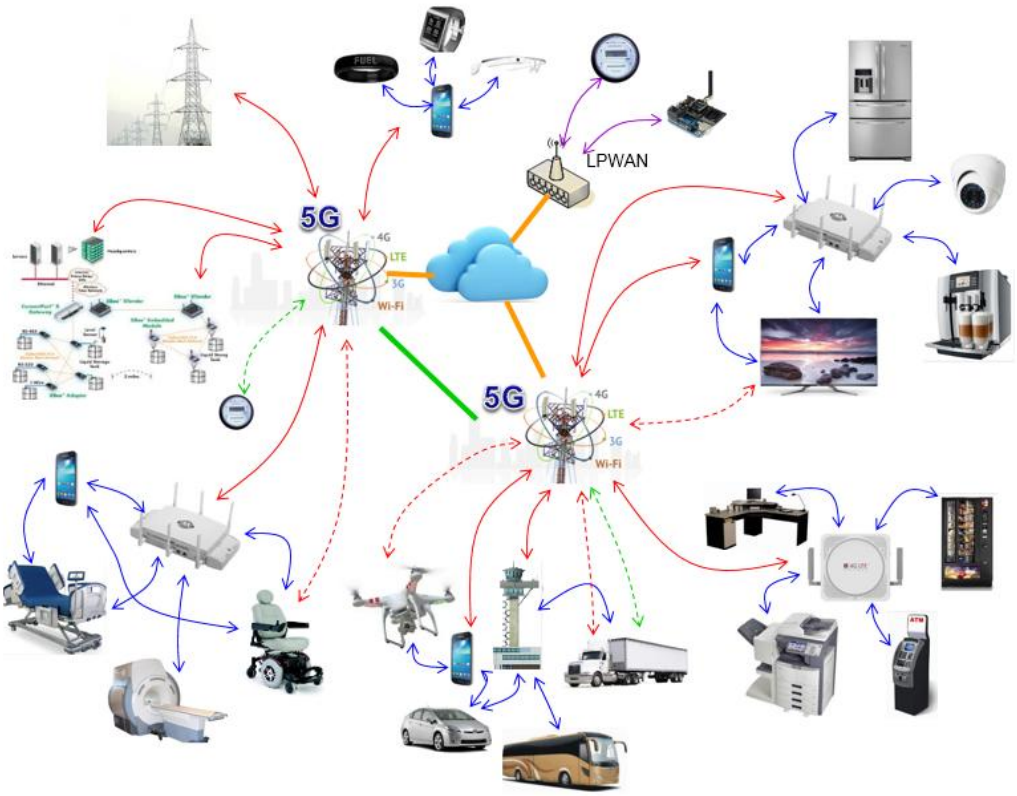
ჭკვიანი მედიცინა - როგორც ექიმებს, ასევე პაციენტებს დისტანციური წვდომა ექნებათ ძვირადირებულ სამედიცინო მოწყობილობასთან და შესაბამისად ავადმყოფობის ელექტრონულ ისტორიასთან. ასევე იქნება რეალიზირებული ჯანმრთელობის დისტანციური მონიტორინგი.

სადღეისოდ, IoT-ის მიმართულებით. ვითარდება მასშტაბური პროექტები ამერიკული სააგენტო "NASA", "Cisco-ს" მხარდაჭერით ქმნის დედამიწის მონაცემთა გლობალურ შეგროვების სისტემას (Planeratary skin) - პლანეტის კანი. ალბათ ბევრს სმენია "Smart house-ზე", სადღეისოდ, იაპონიაში ფართოდ არის გავრცელებული

ჭკვიანი ქარხნები, ხოლო აშშ-ში კი მასიურად მიდის ჭკვიანი ქალაქების პროექტების განვითარება. არა ანაკლები ყურადღება ექცევა IoT-ის განვითარებას ისეთ ქვეყნებში როგორიც არის: ინგლისი, ავსტრალია, იაპონია, სამხრეთ კორეა და სხვა.

IoT-ის ფართო მასშტაბით დანერგვის პროცესში დაბრკოლებას წარმოადგენს სტანდარტიზაციის პრობლემა, ვინაიადან სადღეისოდ, არ არსებობს ერთეული სტანდარტები ნივთების ინტერნეტისთვის, რაც თავის მხრივ ართულებს ინტეგრაციის პროცესს. ასევე ნივთბის ინტერნეტის განვითარებისთვის დაბრკოლებას წარმოადგენს IPv4-დან IPv6-ზე გადასვლა რაც პირველ რიგში არის დაკავშირებული დიდ ხარჯებთან სატელეკომუნიკაციო ოპერატორებისთვის, კერძოდ ქსელური მოწყობილობების მოდერნიზაციის განხორციელებისთვის. ტექნოლოგიური პლატფორმები ნივთების ინტერნეტისთვის ფაქტობრივად შექმნილია, მაგრამ იურიდული და ფსიქოლოგიური ფაქტორები ჯერ კიდევ განვითარების სტადიაშია, ზემოთაღნიშნული პრობლემების მიუხედავად

ნივთების ინტერნეტის ტექნოლოგიაზე გადასვლა გარდაუვალია , იმიტომ რომ ამ ტექნოლოგიას შეუძლია დედამიწის წინ წაწევა. ნახ. 11-ზე იხილეთ IoT-ის ქსელის მაგალითი.



ნახ. 11. IoT-ის ქსელის მაგალითი[7]

წითელი წყვეტილი ხაზით აღნიშნულია კავშირგაბმულობის ხაზი მობილურ ქსელსა და მოწყობილობას შორის. მე-5 თაობაში ეს ლინკი აღინიშნება eMBB და URLLC-ით. ასეთი ტიპის ლინკის ორგანიზება არის რთული, ვინაიდან ის მოითხოვს ძალიან მცირე დაყოვნებას და ექსტამალურად მაღალ სიჩქარეს(HD TV კომპრესიის გარეშე). ასეთი ტიპის ლინკის ორგანიზება წარმოადგენს ერთ ერთ ამოცანას მე-5 თაობის ქსელში. მწვანე წყვეტილი ხაზით აღნიშნულია ლინკი დამაბოლოებელ მოწყობილობებსა და ფიჭურ ქსელს შორის (მაგ.: წყლის მრიცხველი, ენერგიის მრიცხველი). ამ შემთხვევაში ძირითადად გადაიცემა მცირე

გამტარუნარიანობის მონაცემები. ზემოთ აღნიშნული ლინკის ორგანიზება წარმოადგენს LTE Cat 0,1; LTE-M; MTC, NB-LTE-ში მთავარ ამოცანას.

უწყვეტი წითელი ხაზი: ფიჭურდა ქსელსა და მოწყობილობას შორის.

უწყვეტი ლურჯი ხაზით აღნიშნულია: მცირე დიაპაზონში მომუშავე კავშირი მოწყობილობას და გეითვეის შორის. მაგ.:ZigBee, 6 LoPAN და ა.შ.

უწყვეტი მწვანე ხაზი აღნიშნავს სატრანსპორტო ლინკს ფიჭურ საბაზო სადგურსა და ქსელის ბირთს შორის.

იისფერი უწყვეტი ხაზი: LoRa, SigFox.[share]

IoT-ში გამოიყენება ძირითადად შემდეგი ტექნოლოგიები:[7]

- ZigBee
- 6LoPAN
- WLAN
- Bluetooth
- LTN
- Cellular Technology (5G, LTE, WCDMA და ა.შ.)

სადღეისოდ, მკვლევარები მუშაობენ IoT-ის სუსტ მხარეებზე და ცდილობენ მათ გაუმჯობესებას. ოპტიმიზაცია მიმდინარეობს შემდეგი მიმართულელებით:

- ენერგომოხმარების შემცირება
- სენსორული მოწყობილობების ფასის შემცირება
- უსაფრთხოების გაზრდა
- ქსელის მოქნილობის გაზრდა [7]

IoT ქსელის რეალიზაციისთვის არსებობს ტექნოლოგიების რამდენიმე კატეგორია:

1) მოკლე მანძილზე მომუშავე ტექნოლოგიები: ZigBee, Z-Wave, Thread, Bluetooth, WiFi;

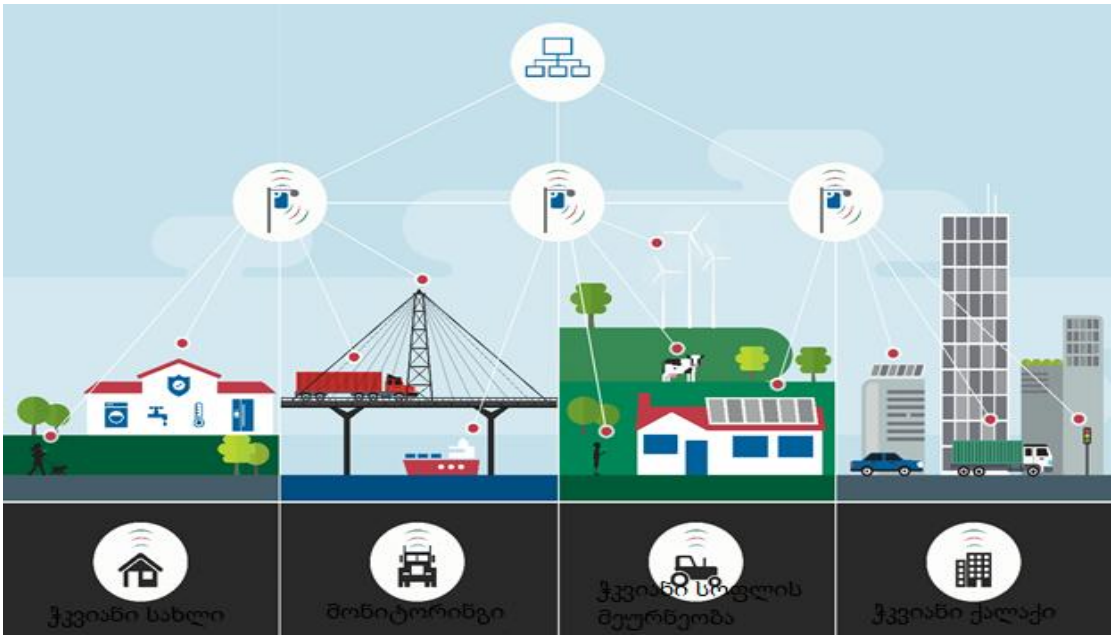
2) ფიჭური: 2G, 3G, 4G (Cat 1; Cat0; LTE-M; NB-IoT), 5G (eMTC,mMTC);
3) LPWN (დაბალი სიმძლავრის უსადენო ქსელი) : Lora; SigFox; NWave; Nuel; Weightless; [7]

IoT-ი წარმოადგენს დონეებიან სტრუქტურას, რომელიც აერთიანებს სხვადასხვა ტიპის ტექნოლოგიებს, თითოეულ დონეზე გარკვეულად განსხვავებული პროცესები მიმდინაორებს , მაგრამ ერთიანობაში ისინი ქმნიან ჰეტეროგენულ ქსელს, ანუ გვაქვს ტექნოლოგიის მონოგენური ქვესიმრავლე; მაგალითად LoRaWAN ტექნოლოგია, და გვაქვს სხვა მონოგენური ქსესიმრავლეები მაგ: Zig Bee, BLE, NFC, WiFi. ამ ქვესიმრავლეების გაერთიანებით საბოლოო ჯამში მიიღება ჰეტეროგენული, ანუ არაერთგვაროვანი ქსელი. ჰეტეროგენულობა გულისხმობს იმას რომ მომხმარებლის სერვისით უზრუნველყოფაში მონაწილეობს იღებენ ზემოთაღნიშნული ტექნოლოგიები ერთი მიზნის უზრუნველსაყოფად. ვინაიდან ჰეტეროგენიზაციის გამოყენება ქსელურ ეფექტურობას საშუალოად ზრდის 30 პროცენტით. ქსელური ეფექტურობა დამოკიდებულია საწყის სურათზე, კერძოდ, ტერიტორიის არაერთგვაროვნება როგორია ტრაფიკის პენეტრაციის კუთხით. ამიტომ მიზანშეწონილია გამოვიყენოთ NDN, არსებული ინტერნეტის და სხვადასხვა ტექნოლოგიების ჰეტეროგენიზაცია. IoT-ში ფართო გამოყენება პოვა LoRa ტექნოლოგიამ.

LoRa - წარმოადგენს LPWAN-ის შემადგენელ ნაწილს. სახელიდან გამომდინარეობს ის, რომ Lora ტექნოლოგიას შეუძლია დიდი ზომის ტერიტორიის დაფარვა, დაბალი სიმძლავრეების გამოყენებით.

ზემოთ აღნიშნული ტექნოლოგიის რეალიზაციის ორი მეთოდი არსებობს: 1) Lora-ს გააჩნია მცირე გაბარიტების ე.წ. “LoRa Gateway” ანუ შეღწევის ბლოკი. LoRa Gateway-ის ანალოგია, რომ მოვახდინოთ მობილურ ქსელებთან, საბაზო სადგურის მსგავსია. თითოეული საბოლოო მოწყობილობა ურთიერთქმედებს Gateway-თან. Gateway უგზავნის

მომხმარებლებს მონაცემებს, ის უხეშად რომ ვთქვათ ასრულებს შუამავლის როლს ქსელის სერვერსა და საბოლოო მოწყობილობას შორის. ზოგადად, მუშაობის პრინციპით გავს მობილურ კავშირს.



ნახ. 12. LoraWAN-ის გამოყენების მაგალითები

აღსანიშნავია, რომ Lora-ს შემთხვევაში არ არის აუცილებელი Gateway-ს გამოყენება, ვინაიდან შეგვიძლია ავაწყოთ ისეთი ქსელი, რომელიც იმუშავებს Gateway-ს გარეშე. პირობითად, ვიყიდოთ სენსორები, ETTUS-ის ფირმის მიმღებ/გადამცემი ან რომელიმე სხვა SDR-ის მოწყობილობა, ჩავიწეროთ უფასო პროგრმული უზრუნველყოფა Linux-ის ბაზაზე, მაგალითად GNU RADIO და ავაწყოთ ისეთი ქსელი, რომელიც იმუშავებს არალიცენზირებულ ISM დიაპაზონში. SDR-ის საშუალებით შესაძლებელი იქნება გარკვეულ დიპაზონში მივიღოთ და გადავცეთ სიგნალი, ხოლო GNU RADIO-ს გამოყენებით კი პროგრამულად დავამუშავოთ სიგნალი. მწარმოებლურობის კუთხით LoRa იყენებს Chirp Spread Spectrum მოდულაციას, რომელიც Zig Bee და BLE ტექნოლოგიებისგან განსხვავედება იმით რომ საშუალებას გვაძლევს

გადავცეთ ქსელში მონაცემები დაბალი სიჩქარით, მაგრამ შორ მანძილზე:[9]

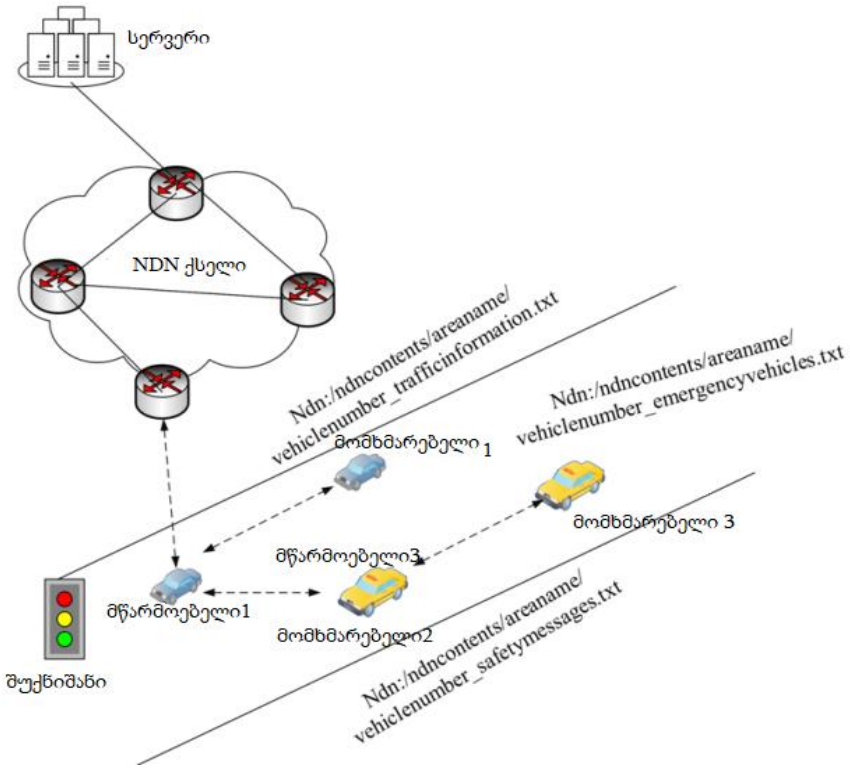
- საშუალოდ გადაცემის მანძილი შეადგენს 15-20 კმ-ს.
- დამაბოლოვებელი მოწყობილობების გადაცემის სიმძლავრე - 40 dBm.
- LoRa მუშაობს ISM დიაპაზონში; Europe – 868 MHz, North America – 915 MHz, Asia – 433 MHz;
- არხის გადაცემის სიგანე: 7,8 KHz; 10,4 KHz; 15,6 KHz; 20,8 KHz; 31,2 KHz; 41,7 KHz; 62,5 KHz; 125 KHz; 250 KHz. პირველი სამი არის ყველაზე პოპულარული.
- ამ ტექნოლოგიის გამოყენებით შეგვიძლია მივაღწიოთ 300 ბტ/წმ -100 კბტ/წმ სიჩქარეს. სიჩქარე იცვლება ადაპტური მოდულაციის შედეგად. აღსანიშნავია, რომ LoRa-ს გამოყენების დროს არ არის აუვილებელი მაღალი სიჩქარეების მიღება, ვინაიდან ის ძირითადად გამოიყენება სენსორებიდან გაზომილი მახასიათებელების გადასაცემად. ნახ.12 იხილეთ LoraWAN-ის გამოყენების მაგალითები.

1.4.1 IoT ქსელის ჰეტეროგენიზაციის სპეციფიკური მაგალითები. სპეციფიკური მაგალითები

Smart City-ს შემთხვევაშიც შეგვიძლია გამოვიყენოთ NDN ქსელი, მაგალითად პიკი საათის დროს, ჭკვიან პარკირებაზე, როდესაც თავისუფალი ადგილის ალბათობა მცირეა, არ არის აუცილებელი ყოველ წამს ინფორმაციის განახლება და დატვირთული მოწყობილობების კიდევ უფრო დატვირთვა. ხელსაყრელია დაქეშირებული ინფორმაციის გაზიარება აბონენტებთან.

ქსელის ოპტიმიზაციის კუთხით შესაძლებელია სატრანსპორტო NDN-ის გამოყენება, რომელიც დაფუძნებულია უსადენო დინამიურ და თვითორგანიზებულ ქსელის პრინციპზე ე.წ, “ad hoc wireless” ასეთი ტიპის

ქსელის კონფიგურაციას ეწოდება VNDN. მოძრავი აბონენტები ქმნიან ქსელს, და ცდილობენ ინფორმაციის გაცვლას დინამიურ რეჟიმში. მაშინ როდესაც სატრანსპორტო საშუალება ჩართულია ქსელში ან ინფრასტრუქტურაში ის შეიძლება ასრულებდეს მონაცემთა მომხმარებლის, წარმოქმნელის , გადამტანის როლს[10]. ნახ. 13-ზე იხილეთ VNDN ზოგადი სქემა.



ნახ. 13. VNDN ზოგადი სქემა

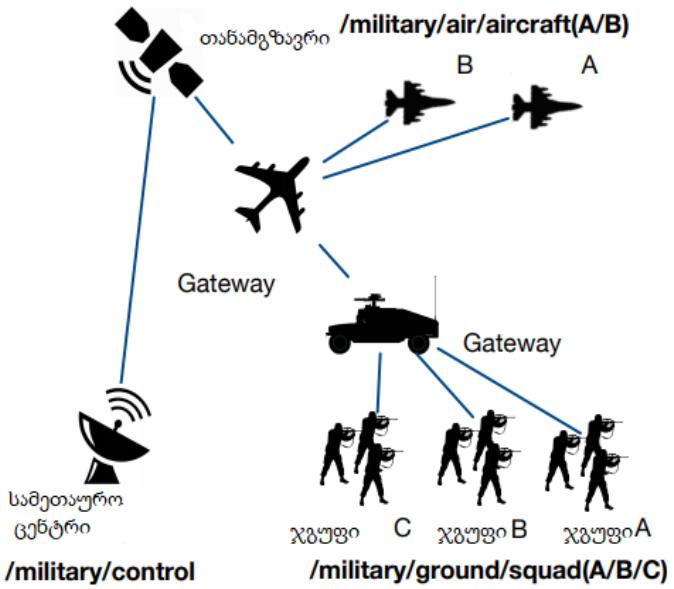
NDN ქსელი წარმოადგენს ქსელის ერთ – ერთ საუკეთესო არქიტექტურას, რომლის თანახმად ქსელის აბონენტი მოთხოვნის საჭიროების შემთხვევაში მოთხოვნას არ აგზავნის წინასწარ განსაზღვრულ IP მისამართზე, არამედ ცდილობს მოძებნოს კონტენტი ქსელში გაგზავნილი ე.წ. ინტერესების საშუალებით. [10]

NDN ქსელში სატრანსპორტო საშუალებების ჩართვის აუცილებლობა განპირობებული იყო 3 ძირითადი მომენტით:

1) ტრაფიკის ინფორმაციის შესახებ ცოდნით: საცობები.
2) დეტალური ინფორმაციით გზის დაზიანების ან მშენებლობის შესახებ.
3) ინფორმაციის გაგებით სასწრაფო დახმარების, პოლიციის, სახანძრო მანქანების შესახებ, რათა მოხდეს ალტერნატიული მარშრუტის არჩევა და შესაბამისად ხელი არ შევუშალოთ სპეცოპერაციების შესრულების დროს.

ნახ.13-ზე ნაჩვენებია V2VNDN ქსელის არქიტექტურა, რომელზეც ჩანს, როგორ აგზავნის მოთხოვნას მომხმარებელი 1, ხოლო მწარმოებელი 1, ანუ სატრანსპორტო საშუალება რომელმაც პირველმა მიიღო ქსელში შემოსული მოთხვონა მიღებისთანავე ჩაიხედავს PIT ინტერესთა მოლოდინის ცხრილში-ში, თუ ასეთი მოთხოვნა უკვე იყო შემოსული მაშინ შეამოწმებს მეორე ცხრილს FIB-ს ინფორმაციის გადაცემის ბაზა-ს, დაკავშირების შემდეგ ამოიღებს მარგი ინფორმაციის ტვირთს CS შემნახველი საცავიდან. იმ შემთხვევაში თუ მოთხოვნილი ინფორმაციის მოძიება ვერ მოხერხდება მაშინ მწარმოებელი 1 გადასცემს ამ ინფორმაციას ქსელში, გააგრძელებს ინტერესს, და ასე გაგრძელდება იქამდე, სანამ რომელიმე მოწყობილობა არ გამოეხმაურება დადებითი პასუხით. ხოლო მწარმოებელი 1 შეინახავს თავის ცხრილში შემოსულ ინტერესს. ჭკვიანი პარკირების შემთხევაშიც მიზენშეწონილი იქნება V2VNDN-ის გამოყენება. [8]

V2VNDN ქსელის გამოყენება განსაკუთრებით მომგებიანია პიკის საათებში როდესაც ბევრი მანქანაა, ინფორმაციის მიწოდება აბონენტამდე უფრო სწრაფია და ეფექტური TCP/IP პროტოკოლთან შედარებით.

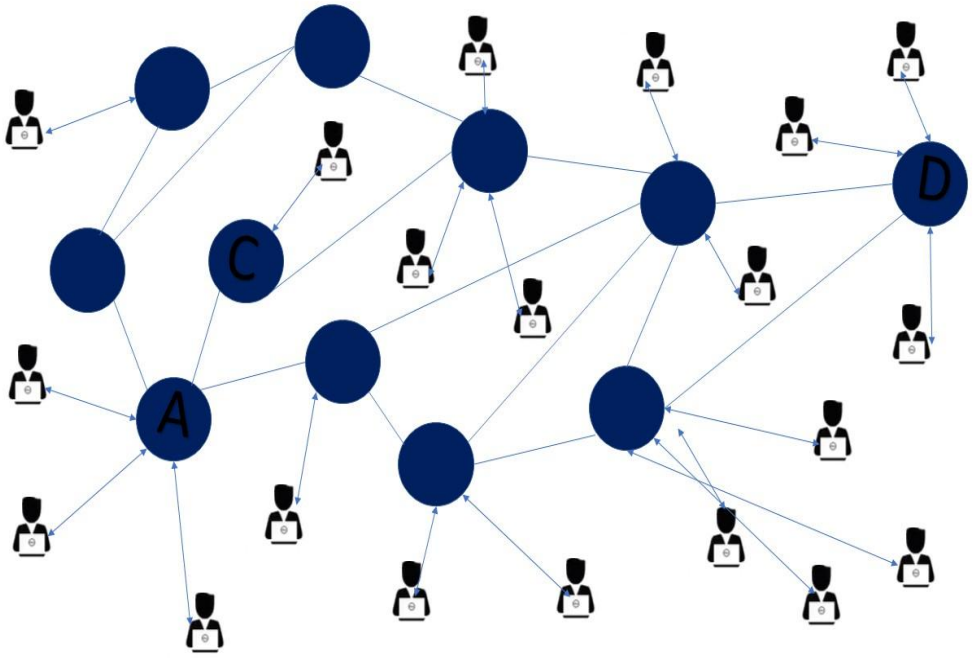


ნახ. 14. NDN-ის გამოყენება ომის პირობებში[1]

21-ე საუკუნეში როდესაც მსოფლიოში ვითარება გამძაფრებულია. მე ვფიქრობ, რომ ყველაფრისთვის უნდა ვიყოთ მზად, და ამისთვის უნდა გვქონდეს შექმნილი ინფრასტრუქტურა, რომ ომის პირობებში კი არ მოგვიწიოს ახალ სისტემაზე და ტექნოლოგიაზე გადასვლა არამედ ამისთვის წინასწარ ვიყოთ მზად და საჭიროების შემთხვევაში გადავერთოთ ახალ სისტემაზე ეტაპობრივად. ომის პირობებში სტრატეგიულად სწორია P2P კონფიგურაციის გამოყენება. არსებული ქსელის საშუალებით ჩვენ ვერ ვუზრუნველყოფთ ინფორმაციის ეფექტურად გამოყენებას, სწორედ ამიტომ გვჭირდება NDN-ის ქსელი, ვინაიდან ის არის უფრო საიმედო ქსელი უსაფრთხოების თვალსაზრისით. აღსანიშნაცია რომ ომის პირობებში შესაძლოა დაიკარგოს კავშირი პუნქტებს შორის და მომხმარებლები ვეღარ დაუკავშირდნენ ერთმენთს. NDN-ის შემთხვევაში კავშირის არხების დაკარგვა იმდენად არ იქნება კრიტიკული. ვინაიდან ინფორმაციის ნაწილი იქნება შენახული, დაქეშირებული. ამიტომ საჭიროების შემთხვევაში სხვადასხვა პუნქტიდან ომის პირობებშიც კი შესაძლებლობა გვექნება იმ ინფორმაციასთან წვდომა რომელიც იყო დროებით შენახული. ხოლო ამ

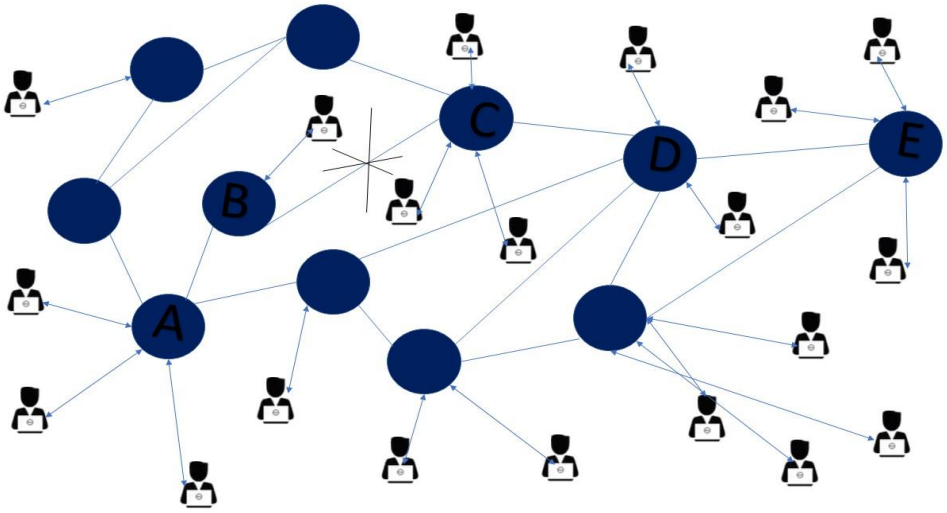
ინფორმაციის მომხმარებლამდე მიტანა მოხდება უფრო სწრაფად ვიდრე არსებულ ქსელში, ვინაიდან ის ინფორმაცია რომელსაც მოითხოვს აბონენტი ნაკლებ მანძილს გაივლის,რაც თავის მხრივ შეამცირებს ქსელში არსებულ დაყოვნებას. ნახ.14. იხილეთ NDN-ის გამოყენება ომის პირობებში.

NDN-ის უპირატესობები ომის პირობებში: შესაძლოა დავწეროთ ისეთი ალგორითმი, რომლის თანახმად როუტერი ინფორმაციის მიღებისთანავე შეინახავს მას ე.წ. CS-ში შემდეგ კი გაავრცელებს ამ ინფორმაციას მულტიქასტით სამეზობლო მოწყობილობებზე(PIT), ანუ გააგრძელებს ინტერესს, შემდგომ სხვა მოწყობილობებიც მოიქცევიან ანალოგურად, მიღებისთანავე ამ ალგორითმის მიხედვით გაავრცელებენ ინფორმაციას ქსელში, ამის დადებითი მხარე არის ის რომ მომხმარებლების გარკვეულ ჯგუფს საკმაოდ სწრაფად შეეძლებათ დაქეშირებული ინფორმაციის ქსელიდან მოპოვება .

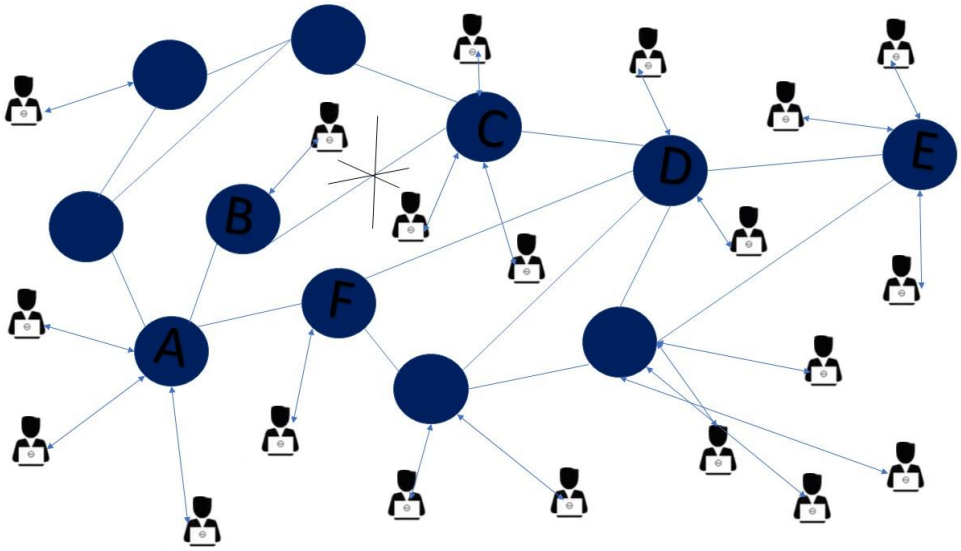


ნახ. 15. NDN ქსელის მაგალითი 1

შესაძლოა მომხმარებლის მიერ გაგზავნილ მოთხოვნაზე წამოღებული ინფორმაცია გზაში დაიკარგოს, სამაგიეროდ დაკარგვის მომენტამდე იქნება შენახული რომელიმე სხვა მოწყობილობაზე, და ამიტომ საშუალება გვექნება გავაგზავნოთ ჩვენი მოთხოვნა კიდევ ერთხელ რათა მივიღოთ ინფორმაცია, რომელიც ვერ მივიღეთ პირველი მოთხოვნის შემდეგ.



ნახ. 16. NDN ქსელის მაგალითი 2



ნახ. 17. NDN ქსელის მაგალითი 3

ნახ. 15-ზე და 16-ზე მოყავნილია მაგალითები: E პუნქტიდან გამოგზავნილი მოთხოვნა A პუნქტში იზგავნება შემდეგი მარშრუტით: ABCDE; შესაძლოა რომ BC ტრანზიტულ გზაზე კავშირი გაწყდეს და ამიტომ NDN-ის შემთხვევაში ხელახლა გაგზავნილი მოთხოვნა A პუნქტიდან E პუნქტის მიმართულებით აღარ მიდის საბოლოო წერტილამდე, ვინაიდან პირველი მოთხოვნის შემთხვევაში წამოსული ინფორმაცია იყო შენახული D პუნქტში, ამიტომ კავშირის გაწყვეტის შემთხვევაშიც კი BC მონაკვეთზე, ხელახალი მოთხოვნის დროსაც კი უმოკლეს მანძილს გაივლის, რაც შეამცირებს გარკვეულწილად დაყოვნებას. აგრეთვე შეგვიძლია შევიმუშაოთ ახალი ალგორითმი, რომლის დროსაც ინფორმაციული პაკეტების დაკავრგვის შემთხვევაში მომხმარებელს არ უწევს ხელახლა მოთხოვნის გაგზავნა, ჩვენს მიერ შემუშავებული ალგორითმი ავტომატურ რეჟიმში მიიღებს გადაწყვეტილებას და თვითონვე დააგენერირებს ახალ მარშრუტს, რათა მომხმარებელმა მიიღოს პასუხი გაგზავნილ მოთხოვნაზე. ნახ. 17 იხილეთ NDN-ის განახლებული ქსელის მაგალით.

განვიხილოთ მაგალითი: საქართველოში მხოლოდ ერთ ტურისტულ კომპანიას აქვს საჰაერო ბურთებით ფრენის სერვისი. რამოდენიმეჯერ გავიგე რომ ფრენა არ შედგა. ამის მიზეზი კი იყო: ქარი, ან სხვა ბუნების მოვლენა, რომელიც წინასწარ არ იყო ცნობილი პილოტისთვის. აღსანიშნავია, რომ ფრენა ძირითადად დილის 6 საათზეა დაგეგმილი, და ამიტომ ხალხის მობილიზება ადგილზე, კერძოდ პილოტის და განსაკუთრებით ტურისტების, ანუ სერვისის მომხმარებელების არის დიდი დისკომფორტი ფრენის გადადების შემთხევაში. ამიტომ მიზანშეწონილია შედგეს წინასწარი ანალიზი, და ამის საფუძველზე წინასწარ ანუ დისტანციურად მოხდეს გადაწყვეტილების მიღება. LoRaWAN ტექნოლოგიის გამოყენებით ჩვენ შეგვიძლია ავაწყოთ ისეთი ქსელი, რომელიც იმუშავებს NDN-ის ბაზაზე. სენსორების განთავსებით ნატახტრის, ალაზნის ველის მიდამოებში, სწორედ ამ პუნქტებიდან სრულდება ფრენები, გავზომოთ გარკვეული პარამეტრები: ტემპერატურა,

ქარის მოძრაობა, ნიადაგის სინესტე რაც ძალიან მნიშვნელოვანია საჰაერო ბურთით ფრენისას. საზოგადად, ცნობილია რომ საჰაერო ბურთით აფრენა სასურველია მზის ამოსვლამდე, სანამ ნიადაგი ჯერ კიდევ ცივია, ხოლო საჰაერო ბურთის გარსში, კერძოდ კი სანთურაში გამოყოფილი ჰაერი გარემოს ჰაერზე ცხელი უნდა იყოს, იმისთვის რომ შედგეს ფრენა. აქედან გამომდინარეობს ის ფაქტი, რომ ცხელ ამინდში სანტურაში უნდა გამოიყოს გაცილებით დიდი რაოდენობის სითბო მიზიდულობის ძალის დასაძლევად. ასევე ითვლება რომ მზის ამოსვლიდან 2-3 სთ-ის განმავლობაში ფრენა ყველაზე უსაფრთხოა. თბილი ნიადაგის დროს, შესაძლებელია ჰაერში შეიქმნას ე.წ. თერმული ეფექტები, ანუ მზის გამოყოფილი სითბოს ენერგიის პენეტრაცია ჰაერში არის სხვადასხვა, ამიტომ ასეთ დროს საჰაერო ბურთით ფრენა არ არის რეკომენდირებული. ჩვენ მიერ შემოთავაზებულ ვარიანტში ტემპერატურის ნიადაგის და ქარის სენსორები გაზომავენ შესაბამის მახასიათებლებს, რომლების დამუშავებაც მოხდება SDR-ის ბლოკში. ხოლო პილოტის მოთხოვნის საფუძველზე მოხდება ინფორმაციის გადაცემა ქსელში, ასევე შესაძლებელია GSM მოდულის გამოყენებაც ქსელში. მაგალითად, პროგრამული სოფტის GNU RADIO-ს საშუალებით დავწეროთ მარტივი if-else კონსტრუქცია C++ ბაზაზე, შევიმუშაოთ ისეთი ალგორითმი, რომელიც პროგრამულად დაამუშავებს პარამეტრებს და ანალიზის საფუძველზე ავტონომურად და ავტომატურად მიიღებს გადაწყვეტილებას და საჭიროების შემთხვევაში GSM მოდულის საშუალებით განხორციელდება ზარი როგორც პილოტთან ასევე მოგზაურებთან, რომელიც მაღვიძარას როლს შეასრულებს. ქსელის რეალიზაციის ამ გზითაც კი უამინდობის გამო შესაძლებელია გადაიდოს ფრენა, სამაგიეროდ ადგილზე მისვლა არ იქნება აუცილებელი, რაც პრაქტიკაში იქნება ძალიან მოსახერხებელი და უმტიკვნელო როგორც პილოტისთვის, რომელიც უხეშად, რომ ვთქვათ მონიტორინგს გაუწევს სისტემას დისტანციურად, ასევე მომხმარებლებისათვის, ანუ ამ კონკრეტულ შემთხვევაში ტურისტებისთვის. არსებობს კიდევ ერთი

ვარიანტი ამ ქსელის განსახორციელებლად. შეგვიძლია არ გამოვიყეონოთ GSM მოდული, ასეთ შემთხვევაში მახასიათებლები უნდა გადაიცეს არსებული ინტერნეტის საშუალებით ან შესაძლებელია გადაიცეს NDN ქსელის საშუალებით. NDN-ის შემთხვევაში, ანალოგია რომ გავიყვანოთ მობილური კავშირის ქსელებთან, ჩვენი ქსელი შეასრულებს LTE-ში არსებული "Small Cell"-ის როლს. რომელიც მოგვცემს ახალი დამატებითი ტრაფიკის შემოღებას, მაგრამ ეს ტრაფიკი იქნება ლოკალური და არ შეუშლის ხელს ქსელში იმ მონაკვეთებზე სადაც ტრაფიკის კონცენტრაცია არის მაღალი და ამავე დროულად გარკვეულწილად გაათანაბრებს ქსელში ტრაფიკის განაწილებას, და გაზრდის ქსელის გამტარუნარიანობას. ამიტომ ვფიქრობ რომ ასეთი ტიპის მიდგომები უნდა დაინერგოს მასობრივად.

ქსელში მულტიმედიური ტრაფიკის გადაცემამ საგრძნობლად მოიმატა. გამოკითხვის შედეგად დაადგინეს, რომ ინტერნეტის მოხმარების 60%-ს წარმოადგენს ვიდეოკონტენტის მოთხოვნები. საქართველოს მასშტაბით არის სოფლები სადაც ინტერნეტის სიჩქარე არის საკმაოდ დაბალი გეოგრაფიული რთული პირობებიდან გამომდინარე, ვერ ხერხდება საკაბელო ინტერნეტის გაყვანა, ამიტომ ქსელის აბონენტი თავისი ინფორმაციული სურვილების დასაკმაყოფილებლად ვალდებულია გამოიყენოს მობილური ქსელი, რომელიც არ არის საიმედოდა ამავ დროულად დიდ დაყოვნებასთან გვაქვს საქმე ინფორმაციის გადაცემის პროცესში. მოხერხებული იქნებოდა NDN ქსელის გამოყენება ასეთ პირობებში, დაკვირვების შედეგად და სტატისტიკის საფუძველზე შეგვიძლია გავიგოთ პოპულარული კონტენტი გარკვეული აბონენტების ინტერესებიდან გამომდინარე და მოვახდინოთ სწორედ იმ კონტენტის დაქეშირება მოწყობილობებზე NDN-ის სპეციფიკის შესაბამისად. ჩემი აზრით ეს არის სწორი ნაბიჯი და წარმოადგენს ინფრასტრუქტურის განვითარების ერთ-ერთ საფეხურს, ვინაიდან დაყოვნების დრო შემცირდება იმ უბნებზე მინიმუმდე და ამავე დროულად მომხმარებლის ინფორმაციულ მოთხოვნებს დავაკმაყოფილებთ ხარისხიანად.

1.5 ხელოვნური ნეირონული ქსელების და ICN ინტეგრაცია

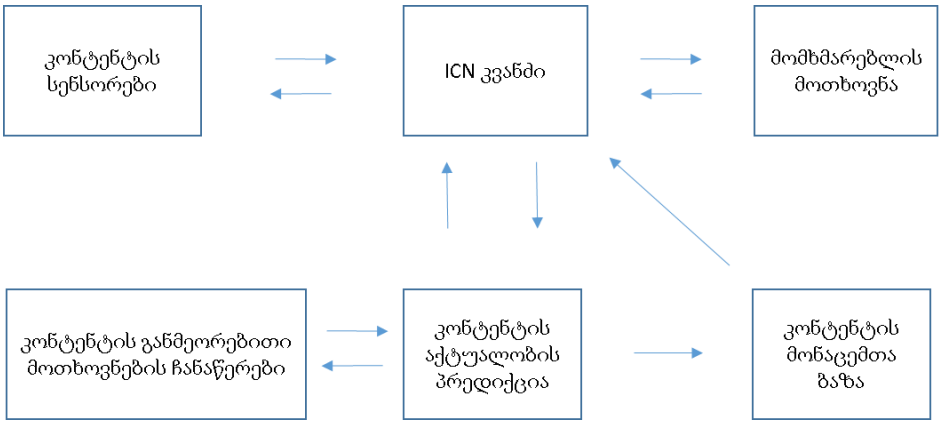
ICN-ის შესაძლებლობა დააქეშიროს მის კვანძზე გამავალი მონაცემები შეზღუდულია მისი შემნახველი დისკის ტევადობით. ამის გამო ასეთ კვანძებს უწევთ პერიოდულად თავისი კონტენტის მონაცემთა ბაზის გადახედვა და მისი დამუშავება. ამის ეფექტური რეალიზაცია მოითხოვს თანამედროვე საშუალებებს.

მიუხედავად იმისა, რომ არსებბული თანმიმდევრული ჰეშირება ძალზე სასარგებლოა, არსებობს გვერდითი მოვლენა, რომელიც გასათვალისწინებელია. როდესაც კონკრეტული ქეში აღებულია ძალიან პოპულარული შინაარსის მომსახურებით, მას შეუძლია განიცადოს მოთხოვნის სიხშირის დონე, რომელიც აღემატება მის რეაგირების შესაძლებლობას. (თუკი მოწყობილობა ვერ ასწრებს ინფორმაციის დამუშავებას მაშინ ინფორმაცია ვერ დაქეშირდება) როდესაც ინდივიდუალური ქეშის დატვირთვა აღემატება კონფიგურაციულ ბარიერს[11].

ხელოვნური ნეირონული ქსელები ასახავენ თავის ტვინის ბიოლოგიურ ნეირონულ ქსელებს. ამასთან, მიუხედავად იმისა, რომ მათემატიკური ალგორითმები კარგად შეეფერება სწორხაზოვან პროგრამირებას, არითმეტიკასა და ლოგიკურ გამოთვლებს, ANN უფრო ეფექტურია პრობლემების გადასაჭრელად, რომლებიც დაკავშირებულია ქაოსური ნიმუშების ამოცნობასთან და შესაბამისობასთან, კლასტერირებასთან და კლასიფიკაციასთან[12].

ჩვენი მოსაზრებით, ნეირონული ქსელის ინტეგრაცია ICN ქსელებში მოგვცემს მაღალი სიზუსტის და ეფექტურობის ალგორითმს, რომელიც შეძლებს კონტენტის წინასწარი აქტუალობის განსაზღვრას. როგორც წესი, სენსორების მონაცემები არ არის დიდი მოცულობის და მათი აქტუალობაც უნდა იყოს სხვა მონაცემებთან შედარებით ფარდობითი. უპირატესბოას ვაძლებთ გენეტიკურ ალგორითმებს, რადგან მათ აქვთ ადაპტაციის კარგი

მექანიზმები. ჩვენ ვთავაზობთ ICN ჭკვიანი ქეშირების სისტემას, რომელიც ნაჩვენებია ნახ.18-ზე. თითო ICN კვანძს ექნება თავისი კონტენტის განმეორებითი მოთხოვნების ჩანაწერები. ეს მონაცემები გაივლის ნეირონული ქსელის შემავალ კვაძებში. ნეირონული ქსელის გამავალი კვანძებიდან მიღებული იქნება მონაცემების აქტუალობის განსაზღვრა და შესაბამისად კონტენტის მონაცემთა ბაზაში ჩასმა ან ამოღება.



ნახ. 18. ICN-ის ჭკვიანი ქეშირების სისტემა

1.6 ინფორმაციაზე ორიენტირებული ქსელის ბაზაზე აწყობილი ქსელი და მისი რეალიზაციის მაგალითი.

მოგეხსენებათ, **ICN** - არის ქსელი, რომელშიც ინფორმაციის მოსაპოვებლად სტანდარტული IP დამისართების ნაცვლად გამოიყენება ე.წ. კონტენტი და ICN ქსელის ერთ ერთი ტიპი არის NDN ქსელი. სადღეისოდ, ზემოთ ხსენებული პლატფორმა პრაქტიკაში არ გამოიყენება, ვინაიდან არის მხოლოდ დამუშავების სტადიაში. NDN-ი ხიბლავს ახალგაზრდა მეცნიერებს, რადგან გააჩნია რიგი უპირატესობები Ipv4-თან შედარებით. არსებულ TCP/IP ქსელში კრიტიკულ პრობლემად ითვლება გარკვეულ ქსელის კვანძებზე გადატვირთვები, რადგან სისტემატურად ხორციელდება ინფორმაციის მრავალჯერადი მოთხოვნა. ამ პრობლემის გადასაჭრელად რაციონალურ ვარიანტად მიგვაჩნია NDN ქსელის

გამოყენება. მობილური კავშირგაბმულობის, კერძოდ 5-ე თაობის განვითარების შედეგად ქსელში IoT მოწყობილობების რაოდენობა კიდევ უფრო გაიზრდება. აღსანიშნავია, რომ IoT სენსორები მოიხმარენ ბატარეის ენერგიას და ამიტომ ერთ-ერთ მნიშვნელოვან საკითხად კავშირგაბმულობის 5-ე თაობის ქსელში მაინც დარჩება IoT მოწყობილობების ენერგომოხმარების ეფექტური გამოყენება. სადღეისოდ, ტენდეცია მიისწრაფის IoT სენსორების მუშაობის ხანგრძლივობის გაზრდისკენ. ამიტომ სწორად მიგვაჩნია ICN-ის ერთ-ერთი სახეობის საშუალებით, კერძოდ NDN ქსელის აგებით განვახორციელოთ ქსელურ უბნებზე გარკვეული კვანძების განტვირთვა და ამის ხარჯზე IoT მოწყობილობების ენერგომოხმარების დაზოგვა. აპრიორი, სენსორებზე შემოსული მოთხოვნების რაოდენობა პირდაპირ პროპორციულია ენერგომოხმარებასთან. მომხმარებლების მიერ სენსორებზე გაგზავნილი მოთხოვნების შემცირებით IoT მოწყობილობები ნაკლებად დაიტვირთვებიან, ვინაიდან არ მოუწევთ ინფორმაციის დამუშავება და შესაბამისად ენერგომოხმარების კუთხით ბატარეის მუშაობის ხანგრძლივობა გაიზრდება.

NDN არის ICN ტიპის ქსელი რომელიც ორიენტირებულია მონაცემებზე. ICN ტიპით შეგვიძლია IoT ენერგომოხმარების და დატვირთვის შემცირება. აღსანიშნავია, რომ თეორიულად Ipv4-ის გაუქმებაც შესაძლებელია, ვინაიდან NDN-ს შეუძლია იფუნქციონიროს, როგორც ცალკეული და დამოუკიდებელი სისტემა. სადღეისოდ, თავსებადობის კუთხით მიზანშეწონილია, რომ დავტოვოთ Ipv4 ქსელი და მოვახდინოთ ინფორმაციული ტრაფიკის გადაცემა ჰეტეროგენული ქსელით.

მოგახსენებთ, რომ Windows ოპერაციული სისტემა დოკუმენტირებული არ არის. Windows-ი Linux-გან განსხვავებით არ არის “open source” და შესაბამისად ქვედა დონეზე ზემოთ აღნიშნულ სისტემის დაპროგრამება და შემდგომ მონაცემების დამუშავება არის ძალიან რთული.

იმის გათვალისწინებით, რომ Linux-ზე უკვე არის მზა ფრეიმვორკი, მე ვფიქრობ რომ უმჯობესია გამოვიყენოთ ეს საშუალება, როგორც ინსტრუმენტარი. ფრეიმვორკი წარმოადგენს ქსელის ბირთვს, რომელზეც იქნება აგებული ICN ტიპის სატესტო ქსელი. ზემოთ ხსენებული ქსელი ჩემს მიერ იყო აწყობილი. ჩვენ დავაყენეთ ფრეიმვორკი და ავაწყეთ, და გავმართეთ ქსელი. გავწერეთ მარშრუტები, გავატარეთ ტუნელები. პირველ რიგში ექსპერიმენტი იყო ჩატარებული Ipv4 ქსელისთვის და შემდგომ NDN-თვის. ბოლო ეტაპზე შევადარეთ Ipv4 და NDN ქსელები, გავაკეთეთ შეფასება.

პითონის დაპროგრამების ენის საშუალებით დავწერეთ სკრიპტი რომლის საშუალებითაც პერიოდულად იგზავნებოდა მოთხოვნა NDN კონტენტზე ლინუქსის ტერმინალის სტანდარტული ბრძანების საშუალებით. სენსორები იყვნენ სიმულირებულნი იგივე პითონის საშუალებით. კონკრეტული პერიოდებით შემთხვევითი ფუნქციით გენერირდებოდა სენსორის ანათვლები და შესაბამისად ძველი ანათვალის აქტუალურობა იყო ვადაგასული, შესაბამისად კონტენტი თავიდან გადიოდა მთელ გზას.

ჩვენს მიერ შემოთავაზებულ ექსპერიმენტში განხილულია თბილისის მასშტაბით აწყობილი ICN ბაზაზე მომუშავე NDN ქსელი. ზემოთ ხსენებული ქსელი აგებული იყო NDN ფრეიმვორკის და შესაბამისი პროგრამული უზრუნველყოფის საშუალებით.

NDN ქსელი აწყობილი იყო კლიენტებისა და სერვერების ერთობლიობით. ამ მოწყობილობებს შორის კავშირის დასამყარებლად გაწერილი იყო NDN ტუნელები, რომლებიც ენკაფსულაციას ახდენდნენ საქართველოს არსებულ ოპერატორების Ipv4-ის ქსელებში. იხილეთ ჩვენი ქსელის ტოპოლოგია.

NDN და IP ქსელების ჰეტეროგენიზაცია, გვაძლევს საშუალებას NDN-ის ბაზაზე შემუშავებული ალგორითმების ხარჯზე შევამციროთ ქსელში ინფორმაციის გადაცემის დაყოვნება.

NDN პაკეტების დამუშავებას ვაკეთებთ NFD(NDN Forwarding Daemon) პლატფორმის საშუალებით. ეს პლატფორმა ჯერ სატესტო ეტაპზეა და მისი საწყისი კოდები თავისუფალ წვდომაშია.

NDN ქსელში პაკეტების მიმოცვლა ხდება ე.წ. ფეისების საშუალებით. ფეისი შეგვიძლია წარმოვიდგინოთ როგორც ფიზიკური ინტერფეისი ან ვირტუალური ტუნელი[13].

NFD არის ფრეიმვორკი, რომელიც ჩვენს ექსპერიმენტში იყო დაყენებული ყველა მოწყობილობაზე, რათა მოწყობილობებს შორის დაგვემყარებინა კომუნიკაცია NDN-ის პროტოკოლების შესაბამისად.

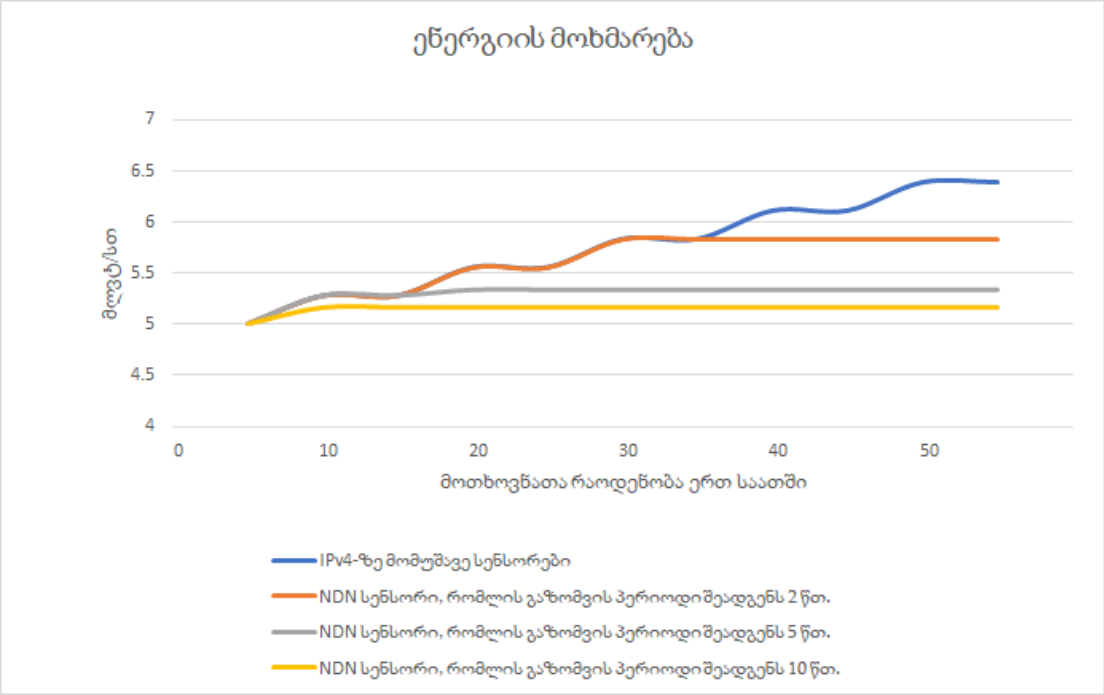
ტუნელები გაწერილი იყო IPv4-ის მისამართებით, კონკრეტულად კი ჩვენს ყველა NDN მოწყობილობებს შორის. როგორც IPv4 ტექნოლოგიას სჭირდება მარშრუტიზაციის პროტოკოლები, ასევე NDN-საც სჭირდება ასეთი. ჩვენს ექსპერიმენტში გამოყენებულია NLSR(Named-data Link State Routing Protocol) მარშრუტიზაციის პროტოკოლი. ამ პროტოკოლით ხდება მარშრუტიზატორებს შორის კონტენტის მისამართების გაზიარება[14].

ეს პროტოკოლი იყო დაყენებული ჩვენს ყველა შუამავალ NDN მოწყობილობაზე. მეზობელ მოწყობილობებს შორის იყო გაწერილი შესაბამისი კონფიგურაცია, რათა ამ მოწყობილობებს შორის დამყარებულიყო მარშრუტების გაზიარების საშუალება.

ექსპერიმენტი:

ჩვენს მიერ აწყობილი სიმულირებული ქსელი იღებს ანათვლებს ყოველ რამდენიმე წუთში ერთხელ. თუკი კლიენტის მიერ მოთხოვნილი ინფორმაციის გადაცემა პირველად ხორციელდება ქსელში, მაშინ როგორც ცნობილია NDN ქსელის ბუნებიდან გამომდინარე კლიენტის ტერმინალიდან გაგზავნილი მოთხოვნა გაივლის მთლიან მარშრუტს. აღსანიშნავია, ის რომ კლიენტის ტერმინალსა და ინფორმაციის შემნახველ სერვერს შორის NDN-ის ბაზაზე მომუშავე ყველა კვანძზე სერვერიდან გამოგზავნილი მონაცემები შეინახება და შესაბამისად ამისა სხვა კლიენტის მიერ მოთხოვნილი მსგავსი ინფორმაციის დროს, მისი ინფორმაციული ტრაფიკის დაკმაყოფილება მოხდება უახლოესი NDN კვანძიდან.

გარკვეული პერიოდის შემდეგ ძველი ინფორმაცია იშლება კვანძებიდან და შემდგომ კვლავ ახლდება. ინფორმაციის ვადაგასულობა დინამიური არგუმენტია, რომელსაც უთითებს ქსელის დამპროექტებელი. ჩვენს მიერ განხილულ ექსპერიმენტში გატესტილია NDN ქსელში სიმძლავრის მოხმარება სხვადასხვა სენსორებისთვის და მათი შედარება IPv4-ის ქსელთან. ქსელში ჩართული გვქონდა სენსორები, რომლებიც გაზომვებს აკეთებდნენ ყოველ 2,5 და 10 წუთში და თითოეული სენსორის მოხმარების სიმძლავრე საშუალოდ უდრის 5 მლვტ/სთ ე.წ.თავისუფალ რეჟიმში, როდესაც სენსორი არ არის აქტიურ რეჟიმში და შესაბამისად კლიენტს არ უგზავნის მონაცემებს. აქტიურ რეჟიმში სენსორი დამატებით მოიხმარს საშუალოდ 100 მლვტ/სთ. სენსორები პირობითად შეგვიძლია წარმოვიდგინოთ, როგორც უსადენო მოწყობილობები, რომლებიც პერიოდულად გამოაგზავნიან მოთხოვნილ ინფორმაციას. ნახ. 19-ზე წარმოდგენილია ენერგიის მოხმარების გრაფიკი.



ნახ. 19. ენერგიის მოხმარების გრაფიკი

სენსორები, რომელთა გაზომვის პერიოდი წარმოადგენს საშუალოდ 10 წუთს, მათი ეფექტურობა NDN ქსელში იკვეთება მაშინ როდესაც

სენსორზე მოთხოვნილი ინფორმაცია რიცხობრივად საშუალოდ 7-ის ტოლია ერთი საათის განმავლობაში. ეფექტურობა მიღწეულია NDN ქეშირების ხარჯზე. იმის ნაცვლად, რომ ყოველი ახალი კლიენტისთვის ინფორმაციის მიღება მოხდეს სერვერიდან NDN ქეშირების საშუალებით ყოველი ახალი კლიენტის მოთხოვნილი ინფორმაცია იქნება დაკმაყოფილებული უახლოესი NDN კვანძიდან, რომელზეც იქნება შენახული სასარგებლო ინფორმაცია.

IPv4 შედეგი შეიძლება აღწერილი იყო შემდეგი ფორმულით:

$$P = N + \frac{Tx \times d \times Rf}{3600} \quad (1)$$

სადაც P არის ჯამური მოხმარებული სიმძლავრე მლვტ/სთ; N - ნომინალური მოხმარებული სიმძლავრე მლვტ/სთ; Tx - კომუნიკაციის საშუალო სიმძლავრე მლვტ/სთ; d - კომუნიკაციის საშუალო ხანგრძლივობა სენსორსა და გეითვეის შორის, რომელიც გამოსახულია წამებში; Rf - მოთხოვნათა სიხშირე.

NDN შედეგი შეიძლება აღწერილი იყო შემდეგი ფორმულით:

$$P = N + \frac{Tx \times d \times Min(Mf, Rf)}{3600} \quad (2)$$

სადაც, Mf - სენსორის გაზომვის სიხშირე საათში.

შედეგების შეფასებით მტკიცედ შეგვიძლია ვთქვათ, რომ ICN ტიპის ქსელში IoT სენსორების ჩართვით მივიღეთ რიგი უპირატესობები IPv4-თან შედარებით, კერძოდ:

მოხდა IoT სენსორების განტვირთვა და შესაბამისად გარკვეული ქსელის უბნებზე დატვირთვა შემცირდა, აგრეთვე სენსორებზე ენერგო მოხმარება დაიზოგა, მიუხედავად გაზრდილი მოთხოვნათა რიცხვისა.

I თავის დასკვნა:

აღვწერეთ პრობლემა დაკავშირებული სენსორების განტვირთვასთან NDN ქსელის საშუალებით და მოვახდინეთ შეფასება ICN NDN IoT-სა და Ipv4 IoT ქსელებს შორის. ქსელის ასაგებად გამოვიყენეთ NDN-ის პლატფორმა და ასევე პითონის პროგრამული ენის საშუალებით მოვახდინეთ სენსორების სიმულაცია. ჩვენს სატესტო ქსელში მივაღწიეთ ენერგომოხმარების გაუმჯობესებას დაზოგვის კუთხით.

ჩემს მიერ შესწავლილ იქნა NDN-ის სტრუქტურა, გამოყენების სპეფიციკა. NDN-ი ემსახურება რამდენიმე მიზანს. ის ზედმეტად არ ტვირთავს ძირითად სერვერს, ამავე დროულად ემსახურება ლოკალურად გარკვეული რაოდენობის ქსელის მომხმარებლებს, ანუ სერვერს ართმევს გარკვეული რაოდენობის ტრაფიკს და შესაბამისად გლობალური ქსელის განტვირთვას ცდილობს. ამცირებს ენერგო მოხმარებას.

ახალ სისტემაზე გადასვლა გარდაუვალია, ამიტომ მნიშვნელოვანია არსებული გეზი, რომელიც აღებულია ნივთების ინტერნეტის განვითარების კუთხით, რომელზეც მუშაობენ ისეთი დიდი ორგანიზაციები როგორებიც არის ITU,IEEE. მიზანშეწონილია ახალ სისტემაზე ეტაპობრივი გადასვლა, ამისთვის კი საჭიროა გვქონდეს შესაბამისი ინფრასტრუქტურა.

სადღეისოდ, NDN-ის მიმართ, ქსელის აქტუალურობიდან გამომდინარე გაზრდილია ინტერესი აკადემიური და სამრეწველო კვლევითი საზოგადოების მხრიდან. 30-ზე მეტი ინსტიტუტი იღებს მონაწილეობას კვლევების განხორციელებაში.

თავი II. MiniNDN-ის პლატფორმაზე ჩატარებული კვლევა და შედეგები

2.1MiniNDN-ის მოკლე მიმოხილვა, დაყენების ინსტრუქცია და ჩატარებული კვლევა

ქვემოთ მოყვანილია MiniNDN-ის დაყენების ინსტრუქცია, რომელიც სრულფასოვნად არ არის აღწერილი ინტერნეტში, თუმცა ექსპერიმენტების საშუალებით ჩვენ მივაღწიეთ დასახულ მიზანს.

sudo apt-get update

sudo apt-get update -y

sudo apt install git

sudo apt install build-essential

sudo apt-get install doxygen

sudo apt-get install python3-sphinx

sudo apt-get install -y pkg-config

sudo apt-get install libsqlite3-dev

sudo apt-get install libssl-dev

sudo apt-get install cmake libblkid-dev e2fslibs-dev libboost-all-dev libaudit-dev

git clone https://github.com/named-data/ndn-cxx.git

cd ndn-cxx

sudo ./waf configure

sudo ./waf

sudo ./waf install

sudo ldconfig

cd ..

sudo apt-get install -y libsystemd-dev

sudo apt-get install valgrind

sudo apt-get install -y libpcap-dev

```
git clone https://github.com/named-data/NFD.git
cd NFD
sudo ./waf configure --without-websocket
sudo ./waf
sudo ./waf install
sudo ldconfig
cd ..
git clone https://github.com/named-data/ChronoSync.git
cd ChronoSync
sudo ./waf configure
sudo ./waf
sudo ./waf install
cd ..
git clone https://github.com/named-data/PSync.git
cd PSync
sudo ./waf configure
sudo ./waf
sudo ./waf install
cd ..
git clone https://github.com/named-data/NLSR.git
cd NLSR
sudo ./waf configure
sudo ./waf
sudo ./waf install
sudo ldconfig
cd ..
git clone https://github.com/named-data/ndn-tools.git
cd ndn-tools
sudo ./waf configure --enable-ping --enable-dump
sudo ./waf
```

```
sudo ./waf install
cd ..
git clone --depth 1 https://github.com/NDN-Routing/infoedit
cd infoedit
make
sudo make install
cd ..
git clone https://github.com/mininet/mininet.git
        echo saved here mid_1_1
        echo longer path can be more reliable
cd mininet
        echo "sudo apt install net-tools" can be fishy?
sudo apt install net-tools
sudo ./util/install.sh -a
cd ..
git clone https://github.com/named-data/mini-ndn.git
        echo saved here mid_2(mininet installed from long dir)
        echo "sudo ./mini-ndn/install.sh -i" gives some errors so longer solution instead
cd mini-ndn
sudo ./install.sh -i
cd examples
sudo python mnndn.py
```

Mini-NDN არის Mininet-ის ბაზაზე აგებული ინსტრუმენტარი, რომელიც ეშვება მხოლოდ ლინუქსზე და გვაძლევს NDN ქსელის ემულაციის საშუალებას. Mini-NDN წარმოადგენს უფასო რესურსს, რომელსაც იყენებენ მეცნიერები ემულირებული NDN ქსელის გასატესტად. პროგრამული უზრუნველყოფა გადმოწერილია github-დან. Mini-NDN-ის გასაშვებად გავხსნათ ლინუქსის ტერმინალი და გავუშვათ შემდეგი ბრძანება: sudo mn.

ქვემოთ მოყვანილია ჩემს მიერ დაწერილი სკრიპტი. ამ სკრიპტის გასაშვებად ლინუქსის ტერმინალში უნდა დავწეროთ შემდეგი ბრძანება: sudo python dinamic_host_1_2.py. სადაც dinamic_host_1_2.py არის პითონის ფაილი.

```
from mininet.log import setLogLevel, info
from minindn.minindn import Minindn
from minindn.util import MiniNDNCLI
from minindn.apps.app_manager import AppManager
from minindn.apps.nfd import Nfd
from minindn.apps.nlsr import Nlsr
from minindn.apps.tshark import Tshark #new
if __name__ == '__main__':
    setLogLevel('info')
    Minindn.cleanUp()
    Minindn.verifyDependencies()
    ndn = Minindn()
    ndn.start()
    info(dir(ndn.net.host('a')))
    info('Starting tshark logging on nodes\n')
  tshark = AppManager(ndn, ndn.net.hosts, Tshark, logFolder="./log/",
singleLogFile=False)
    info('Starting NFD on defualt nodes\n')
    nfds = AppManager(ndn, ndn.net.hosts, Nfd, logLevel='TRACE')
info('Starting NLSR on defualt nodes\n')
    nlsrs = AppManager(ndn, ndn.net.hosts, Nlsr,
logLevel='ndn.*=TRACE:nlsr.*=TRACE')
  new11 = ndn.net.addHost('new1')
    new11.params['params'] = {}
```

```
    homeDir_test = '{}/{}'.format('/tmp/minindn', new11.name)
    new11.params['params']['homeDir'] = homeDir_test
    new11.cmd('mkdir -p {}'.format(homeDir_test))
    new11.cmd('export HOME={} && cd ~'.format(homeDir_test))
    nfds.startOnNode(new11)
    atest = ndn.net.get('a')
    new11.intf('new1-eth0').setIP('100.4.4.3',24)
    atest.intf('a-eth2').setIP('100.4.4.2',24)
    nlsrs.startOnNode(new11)
    info('test: ')
    info(dir(nlsrs.apps))
    ndn.net.get('a').cmd('ndnpingserver /ndn/a-site/a/pingtest > ping-server &')
    ndn.net.get('b').cmd('ndnpingserver /ndn/b-site/b/pingtest > ping-server &')
    ndn.net.get('c').cmd('ndnpingserver /ndn/c-site/c/pingtest > ping-server &')
    ndn.net.get('new1').cmd('ndnpingserver /ndn/new1-site/new1/pingtest > ping-
server &')
    ndn.net.get('d').cmd('ndnpingserver /ndn/d-site/d/pingtest > ping-server &')
   ndn.net.get('d').cmd('echo ''test1'' | ndnpoke ndn:/ndn/d-site/d/test1 &')
    ndn.net.get('d').cmd('echo ''test1'' | ndnpoke -v -x 30000 ndn:/ndn/d-site/d/test1
&')
    ndn.net.get('b').cmd('echo ''test1'' | ndnpoke -x 30000 ndn:/ndn/b-site/b/test1
&')
    ndn.net.get('c').cmd('echo ''test1'' | ndnpoke ndn:/ndn/c-site/c/test1 &')
    #c ndnpeek -p ndn:/ndn/d-site/d/test1
 #c ndnpeek -p ndn:/ndn/b-site/b/test1
a nfdc  face  create  remote udp4://100.4.4.3:6363 local udp4://100.4.4.2:6363
#new1 nfdc  face  create  remote udp4://100.4.4.2:6363 local udp4://100.4.4.3:6363
#already exist
#new1 nfdc route add prefix /ndn/a-site/a nexthop 258 #already exist
```

#a nfdc route add prefix /ndn/new1-site/new1 nexthop 269

#new1 nfdc route add prefix / nexthop 258

#d ip route show

#b sudo sysctl net.ipv4.ip_forward=1 #makes host forward ipv4 packets

#a sudo sysctl net.ipv4.ip_forward=1

d ip route add 10.0.0.0/24 via 10.0.0.9

c ip route add 10.0.0.0/24 via 10.0.0.5

a ip route add 10.0.0.8/30 via 10.0.0.2

b ip route add 10.0.0.4/30 via 10.0.0.1

#c ping -c 4 10.0.0.10

#c ndnping -c 4 /ndn/d-site/d/pingtest

#c ndnping -c 4 /ndn/a-site/a/pingtest

MiniNDNCLI(ndn.net)

ndn.stop()

ქვემოთ განხილულია სკრიპტის უმნიშვნელოვანესი ფუნქციები:

c ndnpeek -p ndn:/ndn/d-site/d/test1 და ndn.net.get('c').cmd('echo ''test1'' | ndnpoke ndn:/ndn/c-site/c/test1 &') ბრძანებების გამოყენებით ხელოვნურად შეგვიძლია შევქმნათ პინგის კონტენტი კონკრეტულ კვანძზე და შემდეგ ჩავატაროთ ექსპერიმენტი პინგის ქეშირების ეფექტურობაზე. ამისთვის დაგვჭრიდება ndn-tool-ს დამატებითი ინსტრუმენტარის დაყენება კომპიუტერზე, კერძოდ ndnpeek და ndnpoke.

გამოყენებული ბრძანებები:

a nfdc face create remote udp4://100.4.4.3:6363 local udp4://100.4.4.2:6363 - ბრძანება ქმნის ე.წ. ფეის ორ კვანძს შორის, ფეისი შეგვიძლია წარმოვიდგინოთ როგორც ინტერფეისი.

new1 nfdc route add prefix / nexthop 258 - NDN-ზე მარშრუტის დამატება

b sudo sysctl net.ipv4.ip_forward=1 - ბრძანება გამოიყენება IPv4 პაკეტების გადამისამართებისთვის.

a ip route add 10.0.0.8/30 via 10.0.0.2 - Mini-NDN-ზე სტატიკური მარშრუტის გაწერა IPv4 პაკეტებისთვის.

c ping -c 4 10.0.0.10 - პინგის გაგზავნა IPv4-ზე

c ndnping -c 4 /ndn/a-site/a/pingtest - NDN-პინგის გაგზავნა შესაბამისი კონტენტის მითითებით

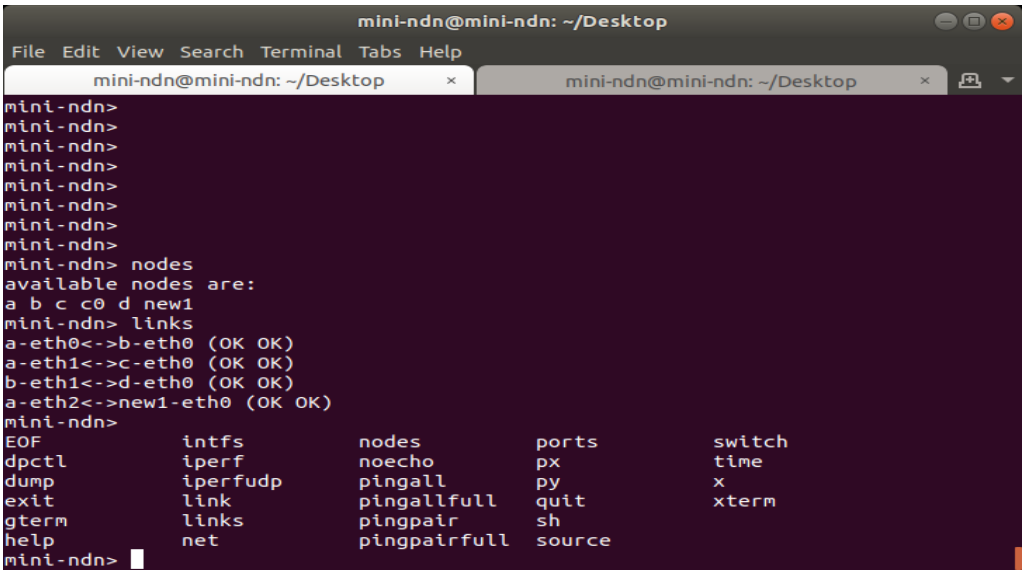
d ip route show - IP მარშრუტების შემოწმება კონკრეტულ კვანძზე.

Mini-NDN-ის ფლატფორმაზე NDN ქსელის განსახორციელებლად ძირითადად მუშაობა გვიწევს Python ან C++ რედაქტორში და ასევე ლინუქსის ტერმინალში. ზემოთ მოყვანილი კოდი ითვალისწინებს ახალი კვანძების შექმნას, კერძოდ: შეიქმნა ახალი დინამიური კვანძი new1; რაც გულისხმობს NFD და NLSR-ის გაშვების შემდეგ ახალი დინამიური new1 კვანძის დამატებას ქსელში.

Mini-NDN-ზე კვანძების შესამოწმებლად ტერმინალში ავკრიფოთ შემდეგი ბრძანება: nodes; ხოლო ლინკების შესამოწმებლად კი - links. ნახ.20 იხილეთ Mini-NDN-ის სამუშაო გარემო.

იმისათვის რომ NDN ქსელმა იფუნქციონიროს ჰოსტებს და როუტერებს უნდა ქონდეთ მხარდაჭერა სახელებზე დაფუძნებული როუტინგის, პაკეტების დამუშავების და ა.შ. სუფთა NDN ბაზაზე აგებული ქსელების გაშვება სადღეისოდ არის არარეალური, ვინაიდან არ არსებობს აპარატურა, რომელიც უზრუნველყოფს მონაცემთა დამუშავებას და გადაცემას NDN პროტოკოლების გათვალისწინებით, აღსანიშნავია რომ NDN-ი არ არის სტანდარტიზირებული და შესაბამისად NDN-ი განვითარების სტადიაშია, მიუხედავად ამისა შესაძლებელია NDN-ის არსებულ სისტემებთან ერთობლივი გამოყენება. NDN-ის პაკეტების UDP/IP ენკაპსულაციის გათვალისწინებით. ეს იმას ნიშნავს რომ NDN-ის ჰოსტიდან UDP/IP მომუშავე როუტერზე მოსული პაკეტი გაიგზავნება შესაბამისი ჰოსტის მიმართულებით, ანუ ის როუტერი რომლისთვისაც არ იყო

განსაზღვრული მონაცემთა პაკეტი, ვერ წაიკითხავს ინტერესთა/მონაცემთა პაკეტის სტრუქტურას, მას უბრალოდ ეცოდინება საით უნდა გააგზავნოს მონაცემები. არსებული ქსელი გარკვეულწილად შეასრულებს მონაცემთა პაკეტის ტრანსპორტირების ფუნქციას.



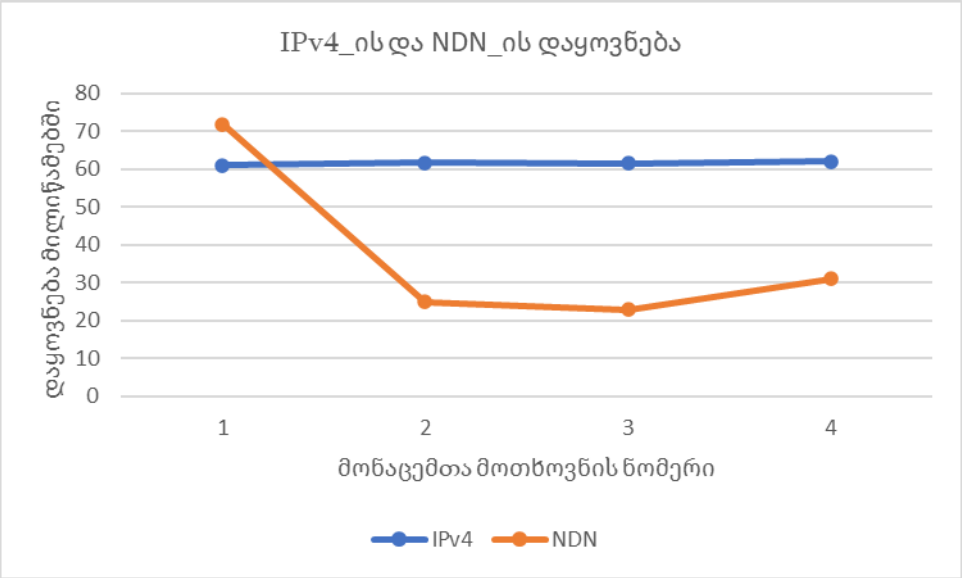
ნახ. 20 Mini-NDN-ის სამუშაო გარემო

ქვემოთ მოყვანილ მაგალითზე ნაჩვენებია, ჩემს მიერ დაწერილი სკრიპტის ფუნქციონალის ტესტირება, კერძოდ დაყოვნების შედარება TCP/IP და NDN ქსელს შორის. პირველ შემთხვევაში გაშვებულია C კვანძიდან კონკრეტულ ფიზიკურ მისამართზე, კერძოდ 10.0.0.10 ოთხი პინგი. მინიმალური დაყოვნების დრო შეადგენს 61.0 მწმ. NDN-ის შემთხვევაში კი 22.883 მწმ. აღსანიშნავია, რომ პირველი პინგის დრო NDN-ის შემთხვევაში შეადგენს 71.9338 მწმ, რაც თითქმის 11მწმ მეტია TCP/IP-თან შედარებით. ეს ფაქტი აიხსნება იმით რომ NDN-ის კვანძზე დაქეშირებული ინფორმაციის დამუშავებას სჭირდება გარკვეული დრო. დაყოვნების პროცენტული შედარება TCP/IP და NDN-ს შორის იხილეთ შემდეგ ნახაზებზე : ნახ.21, ნახ.22, ნახ.23. ქვემოთ მოყვანილია Mini-NDN-ის ტერმინალიდან გაშვებული პინგი TCP/IP და NDN შემთხვევებისთვის.

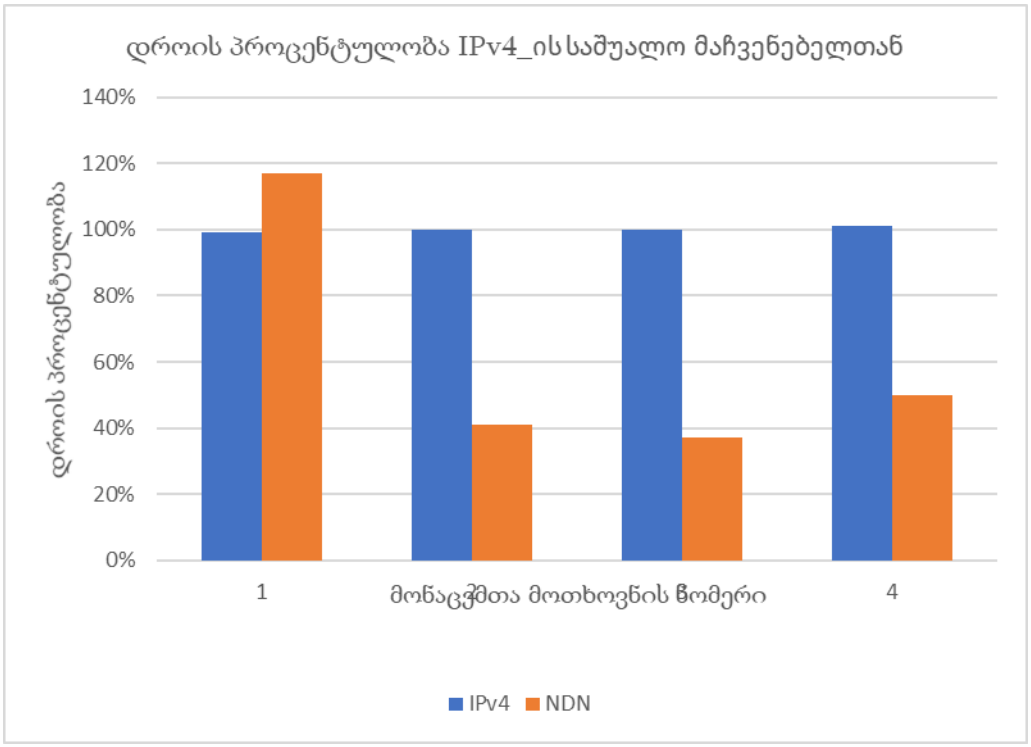
```
mini-ndn> c ping -c 4 10.0.0.10
PING 10.0.0.10 (10.0.0.10) 56(84) bytes of data.
64 bytes from 10.0.0.10: icmp_seq=1 ttl=62 time=61.0 ms
64 bytes from 10.0.0.10: icmp_seq=2 ttl=62 time=61.8 ms
64 bytes from 10.0.0.10: icmp_seq=3 ttl=62 time=61.5 ms
64 bytes from 10.0.0.10: icmp_seq=4 ttl=62 time=62.0 ms
--- 10.0.0.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 61.018/61.631/62.030/0.494 ms

mini-ndn> c ndnping_cache -c 4 /ndn/d-site/d/pingtest
PING /ndn/d-site/d/pingtest
content from /ndn/d-site/d/pingtest: seq=15949988448206744268 time=71.9338 ms
content from /ndn/d-site/d/pingtest: seq=15949988448206744269 time=24.9698 ms
content from /ndn/d-site/d/pingtest: seq=15949988448206744270 time=22.883 ms
content from /ndn/d-site/d/pingtest: seq=15949988448206744271 time=31.0602 ms
--- /ndn/d-site/d/pingtest ping statistics ---
4 packets transmitted, 4 received, 0 nacked, 0% lost, 0% nacked, time 150.847 ms
rtt min/avg/max/mdev = 22.883/37.7117/71.9338/17.11105 ms
```

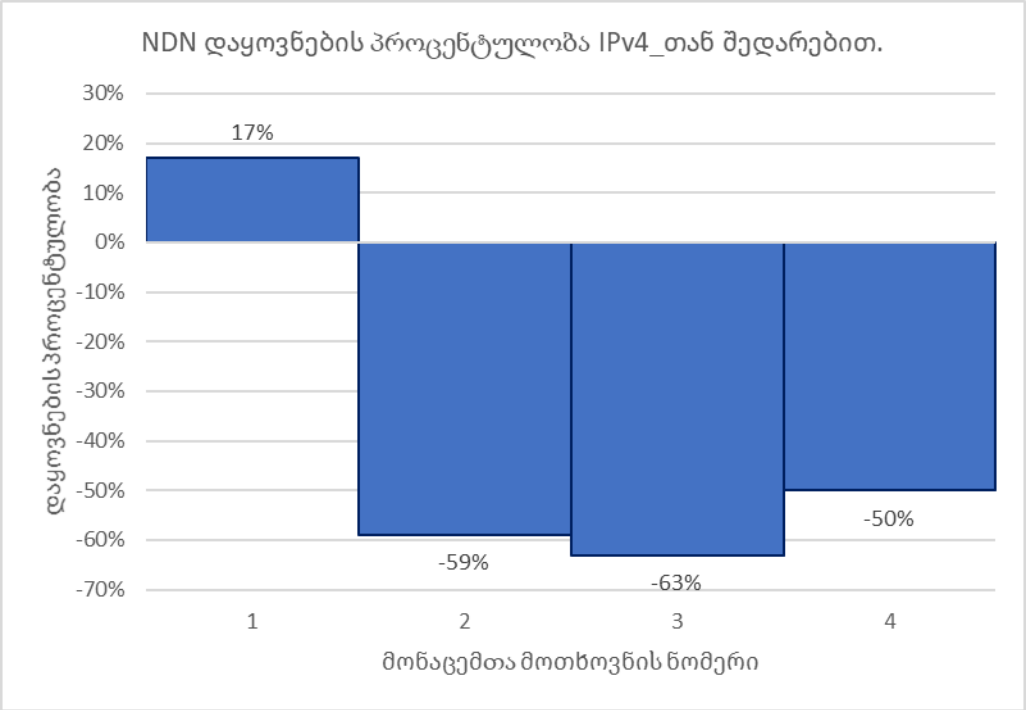
NDN და IP ქსელების ჰეტეროგენიზაცია, გვაძლევს საშუალებას NDN-ის ბაზაზე შემუშავებული ალგორითმების ხარჯზე შევამციროთ ქსელში ინფორმაციის გადაცემის დაყოვნება.



ნახ. 21. IPv4/NDN დაყოვნება



ნახ. 22. დროის პროცენტულობა IPv4 საშუალო მაჩვენებელთან

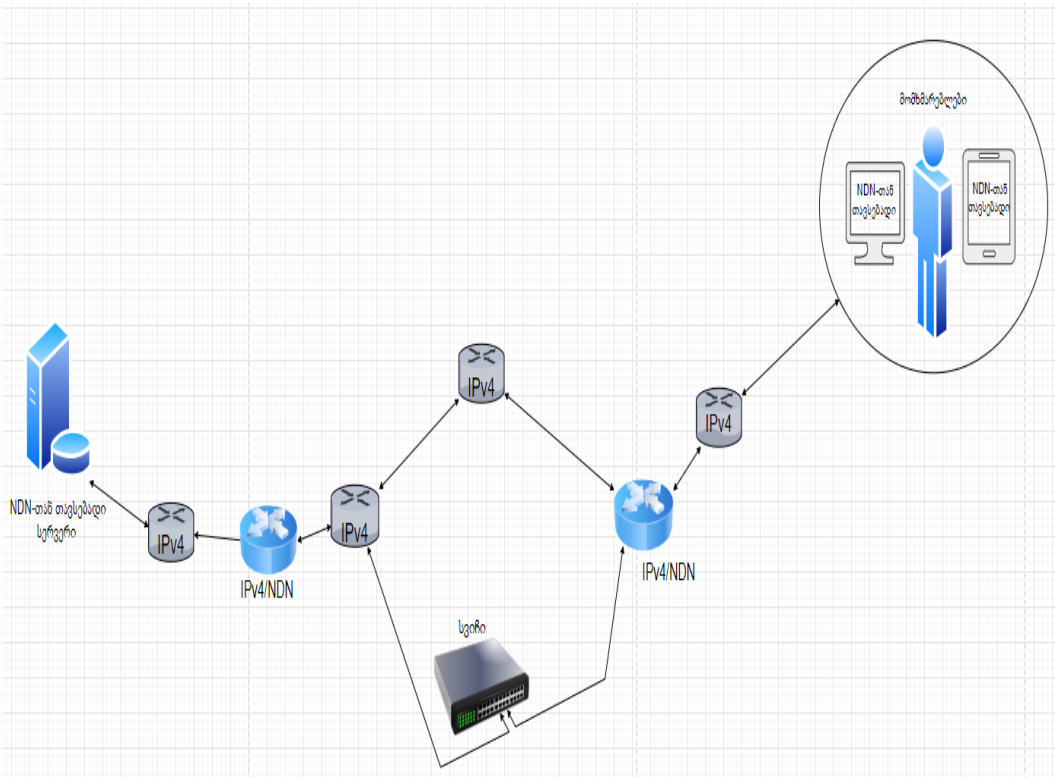


ნახ. 23. NDN დაყოვნების პროცენტულობა IPv4 შედარებით

2.2 NDN ქსელის მაგალითის ტოპოლოგია

ცხადია, NDN მოთხოვნის ქსელში გასაგზავნად ქსელის მომხმარებელს კომპიუტერზე ან ტელეფონზე უნდა ქონდეს დაყენებული აპლიკაცია, რომელსაც ექნება NDN-ის მხარდაჭერა. NDN მოთხოვნის ინფორმაციულ პაკეტში დანიშნულების ადგილის ველში ჩაიწერება NDN სერვერის მისამართი და გაიგზავნება დეფოლთ გეითვეიზე. ARP პროტოკოლის მეშვეობით გავიგებთ IPv4-ზე მომუშავე როუტერის ინტერფეისის MAC მისამართს და შესაბამისად მომხმარებლის მიერ გაგზავნილი ე.წ. ინტერესთა პაკეტი, რომელიც იქნება ენკაფსულირებული IPv4 პროტოკოლით გაიგზავნება IPv4-ზე მომუშავე როუტერის მიმართულებით, შემდგომ მიღებისთანავე როუტერი ჩაიხედება პაკეტში და წაიკითხვას დანიშნულების ადგილს და გადაამისამართებს ამ პაკეტს IPv4/NDN როუტერის მიმართულებით. IPv4/NDN როუტერი ზემოთ აღნიშნული პაკეტის მიღებისთანავე თავისი სამეზობლოს, კერძოდ LSDB

ცხრილის, LSA Name-ის გათვალისწინებით გააგზავნის პაკეტს. ნახ. 24. ნაჩვენებია NDN-ის ტოპოლოგიის მაგალითი.



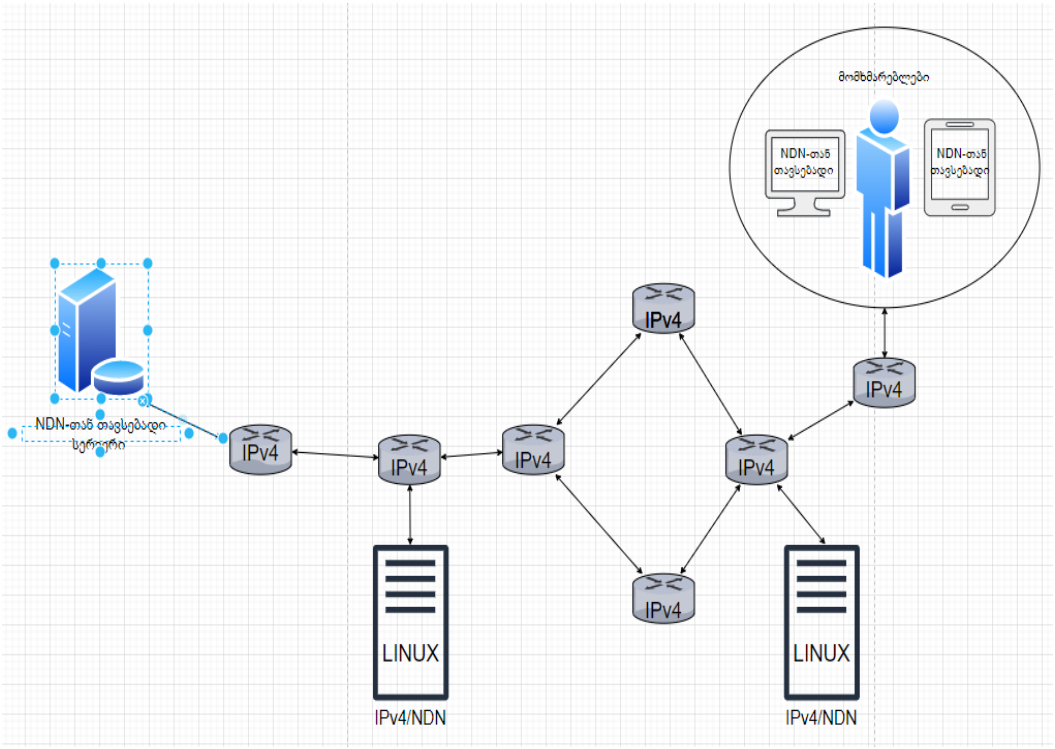
ნახ. 24. NDN-ის ტოპოლოგიის მაგალითი

ARP პროტოკოლის გათვალისწინებით სვიჩი გადაამისამართებს ამ პაკეტს IPv4 როუტერზე, ის ჩაიხედება პაკეტის თავსართში და ნახავს ვისთვის იყო განსაზღვრული ზემოთ აღნიშნული პაკეტი და შემდგომ ისევ გააგზავნის მას ქსელში. ეს პროცედურა გაგრძელდება იქამდე სანამ პაკეტი არ მივა თავის ადრესატამდე. ანალოგიურ გზას გადის ე.წ. მონაცემთა პაკეტი სერვერიდან მომხმარებლამდე. NDN ქსელის სქემა იხილეთ ნახ. 25-ზე

იქიდან გამომდინარე რომ NDN-ი დამუშავების სტადიაშია, NDN ქსელის გაშვება როგორც ცალკე ინფრასტრუქტურის სადღეისოდ, არ არის შესაძლებელი. გარდა ამისა არ არსებობს მოწყობილობები, რომლებსაც ესმით NDN-ის სტრუქტურა, აღნიშნულის მიუხედავად, NDN ქსელის

გაშვება შესაძლებელია თეორიულად UDP ტუნელის საშუალებით. ამისთვის ქსელში უნდა გამოყენებულ იქნას რომელიმე ვარიანტი:

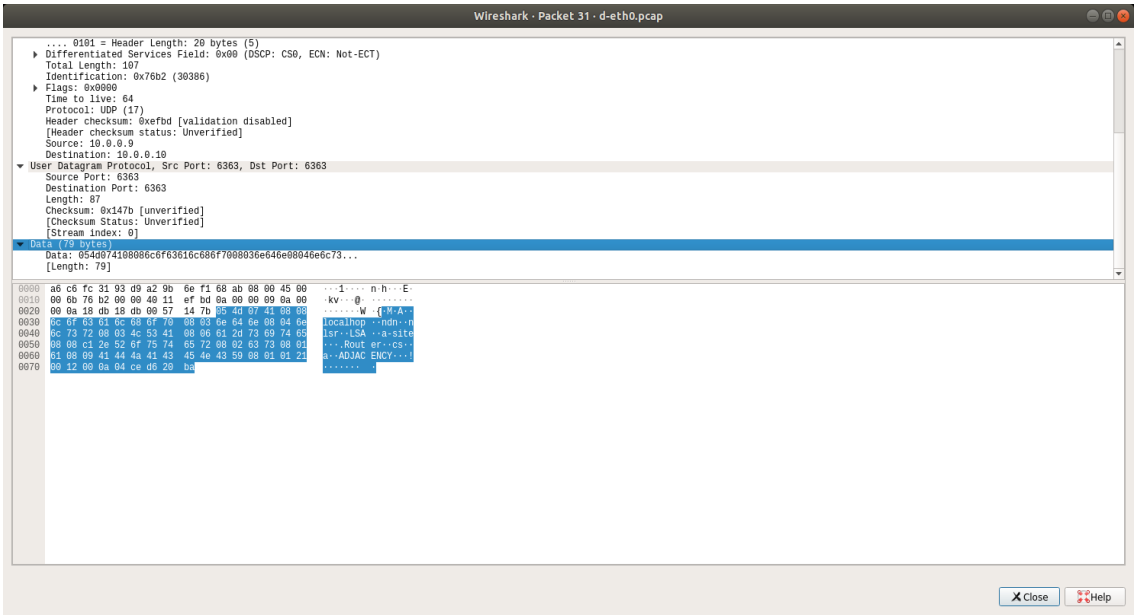
1) მოწყობილობები, რომელთა ოპერაციულ სისტემებზე, კერძოდ ლინუქსზე იქნება დაყენებული NFD და NLSR
2) როუტერები, რომლებზეც ეყენება NFD და NLSR ,ანუ განახლებული პროგრამული უზრუნველყოფით
3) ცისკოს ჭკვიანი როუტერები, რომლებსაც დაემატება ახალი მოდული NFD,NLSR-ის მხარდაჭერით.



ნახ. 25. NDN ქსელის სქემა

NDN-ით დაგენერირებული ე.წ. ინტერესთა/მონაცემთა პაკეტი გაიგზავნება არსებული UDP/IP ქსელით, გაივლს ენკაფსულაციას IPv4-ით, შესაბამისად IPv4/IPv6-ზე მომუშავე როუტერი NDN-ით დაგენერირებული პაკეტების მიღებისთანავე შეხედავს მხოლოდ მიმღების ადრესატს და შემდგომ გადამისამართებს NDN პაკეტს IP დინამიური/სტატიკური

პროტოკოლის გათვალისწინებით. IPv4-ზე მომუშავე როუტერებს არ ექნებათ საშუალება NDN-ის პაკეტების წაკითხვის, რადგან მათ არ ესმით NDN-ის პაკეტების სტრუქტურა. ნახ. 26-ზე მოყვანილია wireshark-ში დაჭერილი NDN პაკეტის სტრუქტურა.



ნახ. 26. Wireshark-ში დაჭერილი NDN პაკეტის სტრუქტურა

NDN მონაცემები გადაიცემა IPv4-ის საშუალებით, და კერძოდ Wireshark-ის პროგრამაში დაჭერილი პაკეტის data-ში შეგვიძლია ვიხილოთ NDN-ის პაკეტი, მაგრამ მოცემული პაკეტი წარმოადგენს ბიტების დაშიფრულ თანმიმდევრობას.

IPv4/IPv6 როუტერები გამტარი მოწყობილობები იქნებიან, ანუ შუალედური მოწყობილობების როლს შეასრულებენ. NDN მოწყობილობებს საჭიროა რომ ქონდეთ დაქეშირების ფუნქცია, იმიტომ რომ ყველა აპარატურას არ აქვს მაგის მხარდაჭერა და შესაბამისად ჩვეულებრივ როუტერებზე დაქეშირების ფუნქციის გამოყენება არსებული TCP/UDP-ით იქნება შეუძლებელი.

ქსელში უნდა იყოს სამი სახის დამისამართება:

1) MAC (სვიჩებისთვის)

2) IPv4 (როუტერებისთვის) - ქსელში NDN/IP პაკეტების გადამისამართებისთვის.

3) NDN (როუტერებისთვის) - NDN პაკეტების მარშრუტიზაციისთვის.

NDN მარშრუტიზაციის პროტოკოლი ავრცელებს მარშრუტიზაციის განახლებებს და ანგარიშობს მარშრუტებს ე.წ. იერარქიული სახელების პრეფიქსებისთვის.

NDN-ის საუკეთესო მარშრუტის გამოსათვლელად შესაძლოა გამოყენებულ იქნას IP-ს ნებისმიერი მარშრუტიზაციის ალგორითმი; მაგ.: ლინკის მდგომარეობის ან სიგრძის ვექტორის. თუმცა NDN მარშრუტიზაციის პროტოკოლს უნდა ქონდეს მრავალგზიანი გადაცემის შესაძლებლობა, ანუ უნდა შეეძლოს მრავალი პუნქტის შემოთავაზება როუტერისთვის ინფორმაციული პაკეტის გადასაცემად, სადაც მრავალგზიანი გადაცემა მიმართულია ინფორმაციის მწარმოებლისკენ ან ერთსა და იმავე ინფორმაციის მქონე მრავალ მწარმოებლისკენ. გარდა იმისა, რომ NDN-ში საჭიროა სახელების პრეფიქსების მისაწვდომობის გავრცელება IP პრეფიქსების ნაცვლად.

NLSR -ს გააჩნია შემდეგი უპირატესობები:

1) სახელები: NLSR-ი იყენებს იერარქიულად სტრუქტურირებულ სახელებს როუტერების ამოსაცნობად, მარშრუტიზაციის პროცესებისთვის და ა შ.
2) უსაფრთხოება: იქიდან გამომდინარე, რომ ყოველი NLSR-ის შეტყობინება იგზავნება NDN მონაცემთა პაკეტის საშუალებით, რომელსაც აქვს ხელმოწერა, როუტერს შეუძლია ამ ხელმოწერის ვერიფიკაცია თითოეული შეტყობინების, ინფორმაციის სწორ ადგილამდე მისატანად .

3) მრავალგზიანი გადაცემა: IP-გან განსხვავებით NDN-ს შეუძლია მრავალგზიანი გადაცემის გამოყენება, ვინაიდან გადაცემის პროცესის ფუნქციაში გათვალისწინებულია ლუპებისგან თავის აცილება.

NLSR სრულყოფილი არ არის, იმიტომ რომ დამუშავების სტადიაშია ეხლა. Neighbors მეზობლები მხოლოდ სტატიკურები არიან, ამიტომ ჩემი მიზანია მომავალში NLSR-ის დამუშავება და კერძოდ დინამურობის შექმნა რაც ითვალისწინების კონფიგურაციის ფაილის გაშვების შემდეგ Mini-NDN-ის ტერმინალიდან ახალი მეზობლების დამატებას. ნახ. 27-ზე მოყვანილია NLSR კონფიგურაციის ფაილი.

ნახ. 27. NLSR კონფიგურაციის ფაილი

კონფიგურაციის ფაილში მეზობლების დამატების მაგალითი:

```
neighbor
  {
    name /ndn/new1-site/%C1.Router/cs/new1
    face-uri udp://100.4.4.3
```

```
    link-cost 10
  }
```

კონფიგურაციის ფაილში შეგვიძლია გავწეროთ მეზობლობა კვანძებს შორის, მაგრამ კონფიგურაციის ფაილის გაშვების შემდეგ მეზობლობის გაწერა კვანძებს შორის Mini-NDN-ის ტერმინალიდან შეუძლებელია NLSR-ის რიგი შეზღუდვების გამო.

NLSR-ში DNS ფუნქციების არ არსებობს გამო ვფიქრობ მიზანშეწონილი იქნებოდა ერთი დინამიური მისამართის გამოყენება მარშრუტების სუმარიზაციისთვის და დანარჩენი მისამართების კი კონტენტის ადგილმდებარეობის განსაზღვრისთვის. კონტენტისთვის განსაზღვრული მისამართები არ შეიცვლება ხოლო დინამიური მისამართები კი მიიღებენ მონაწილეობას მარშრუტიზაციის პროტოკოლებში. ის ჰოსტები, რომლებსაც არ აქვთ კონტენტის შემნახველი ბაზები არ ექნებათ სტატიკური NDN მისამართები, ჩემი აზრით ასეთი მიდგომა ნაკლებად დატვირთავს მარშრუტიზაციის ცხრილებს და ამავე დროულად ლუპებისგან თავს ავიცილებთ. ამიტომ არ არის აუცილებელი, რომ ყველა ჰოსტმა იცოდეს თავისი სამეზობლო, და ამაზე დახარჯოს რესურსები.

2.3 NLSR უსაფრთხოება

NDN-ის მონაცემთა პაკეტის ერთ-ერთი შემადგენელი ნაწილია ე.წ. ციფრული ხელმოწერა. ხელმოწერა შეიცავს: სახელს, კონტენტს, მეტადატას ციფრული ვერიფიკაციისთვის . მეტადატასი ერთი ნაწილი წარმოადგენს გასაღების მომპოვებელს, რომელიც მიუთითებს გასაღების სახელწოდებას და გამოიყენება პაკეტის ხელმოსაწერად, მიმღების მხარის ხელმოწერის ვერიფიკაციისთვის. მიმღებს უნდა შეეძლოს გასაღების ვერიფიკაცია რათა შემდგომ ხელი მოაწეროს LSA-ს პაკეტის, ჭეშმარიტი მოდელის გასაღების აუტენტიფიკაციას.

ქსელი შედგება მრავალი საიტისგან თითოეულ საიტს გააჩნია ერთი ან მეტი ოპერატორი, რომელიც მართავს საიტის როუტერებს. თითოეულ როუტერს შეუძლია შექმნას NLSR პროცესი, რომელიც მართავს LSA-ს. LSA უნდა იყოს ხელმოწერილი მოქმედი NLSR-ით , იმავე როუტერზე სადაც LSA იყო შექმნილი. იმისათვის რომ NLSR პროცესი გახდეს მოქმედი, გასაღების პროცესი უნდა იყოს ხელმოწერილი შესაბამისი როუტერით, რომელიც თავის მხრივ უნდა იყოს ხელმოწერილი რომელიმე საიტის შესაბამისი ოპერატორით. თითოეული საიტის ოპერატორის გასაღები უნდა იყოს ხელმოწერილი საიტის გასაღებით, რომელიც თავის მხრივ უნდა იყოს სერტიფიცირებული ქსელის ძალაუფლებით, ქსელის გასაღების გამოყენებით ე.წ. ნდობის ბმულით. გასაღებებს შორის ნდობის კავშირის კონფიგურაციის ფაილი ნაჩვენებია ნახ.28-ზე.

```
security
{
  validator
  {
    rule
    {
      id "NSLR LSA Rule"
      for data
      filter
      {
        type name
        regex ^[^<NLSR><LSA>]*<NLSR><LSA>
      }
      checker
      {
        type customized
        sig-type rsa-sha256
        key-locator
        {
          type name
          hyper-relation
          {
            k-regex ^([^<KEY><NLSR>]*)<NLSR><KEY><ksk-.*><ID-CERT>$
            k-expand \\1
            h-relation equal
            p-regex ^([^<NLSR><LSA>]*)<NLSR><LSA>(<>*)<><><>$
            p-expand \\1\\2
          }
        }
      }
    }
    rule
    {
      id "NSLR Hierarchy Exception Rule"
      for data
      filter
      {
        type name
        regex ^[^<KEY><%C1.Router>]*<%C1.Router>[^<KEY><NLSR>]*<KEY><ksk-.*><ID-CERT><>$
      }
      checker
      {
        type customized
        sig-type rsa-sha256
        key-locator
        {
          type name
          hyper-relation
          {
            k-regex ^([^<KEY><%C1.Operator>]*)<%C1.Operator>[^<KEY>]*<KEY><ksk-.*><ID-CERT>$
            k-expand \\1
            h-relation equal
            p-regex ^([^<KEY><%C1.Router>]*)<%C1.Router>[^<KEY>]*<KEY><ksk-.*><ID-CERT><>$
            p-expand \\1
          }
        }
      }
    }
    rule
    {
      id "NSLR Hierarchical Rule"
      for data
      filter
      {
        type name
        regex ^[^<KEY>]*<KEY><ksk-.*><ID-CERT><>$
      }
      checker
      {
        type hierarchical
        sig-type rsa-sha256
      }
    }
    trust-anchor
    {
      type file
      file-name "root.cert"
    }
}
```

ნახ. 28. ნდობის კავშირი გასაღებებს შორის

მომიჯნავე კვანძები ერთმანეთში მოლაპარაკების შედეგად აღდგენენ გასაღებს NLSR-ის მეშვეობით. აღდგენის გასაღები: NDN-ში, ღია გასაღების

მონაცემებია, რომლებიც მიიღწევა ინტერესთა/მონაცემთა პაკეტების გაცვლით. მეზობლები ეძებენ მოთხოვნილ გასაღებს კონტენტის საცავში(CS). თუკი გასაღები არ მოიძებნა მაშინ ხელახალი მოთხოვნა იქნება გაგზავნილი. გასაღების აღსადგენად საჭიროა სახელის პრეფიქსის მითითება, კერძოდ /hnetworki/broadcast/KEYS, რომელიც მოთხოვნილი გასაღების წინ იწერება. თითოელი როუტერი ავრცელებს ქსელში ბროუდქასთს ,გასაღების მოთხოვნის მისაღებად. როუტერების პასუხი გასაღების მოთხოვნაზე წარმოდგენილია ენკაფსულირებული მონაცემთა პაკეტის სახით, რომელიც შეიცავს მოთხოვნილ გასაღებს. აღსანიშნავია, რომ გასაღების აუტენტიფიკაცია დამოკიდებულია შიდა პაკეტის მონაცემთა ხელმოწერაზე, გარე პაკეტის ხელმოწერის ველი შევსებულია SHA-256 ჰეშით და გამოიყენება პაკეტის შემოწმებისთვის.

2.4 ემულირებულ NDN ქსელში ხელმოწერილი პაკეტების მიერ გაზრდილი გამოთვილითი დატვირთვის განსაზღვრა

ჩვენ ჩავატარეთ პატარა ექსპერიმენტი Mini-NDN პლატფორმის საშუალებით. ამის მიზანი იყო, გასაღების აუტენტიფიკაციის არსებობის და არ არსებობის პირობებში, შეგვეფასებინა CPU დათვირთვა. ტოპოლოგიაში აღებული იყო 20 NDN კვანძი და მათი დამაკავშირებელი 40 ლინკი. ექსპერემენტი დაყოფილი იყო 4 ეტაპად, აქ ხდებოდა NLSR_ის ყველაზე ინტენსიური შეტყობინებების მიმოცვლა. თავდაპირველად NLSR გაშვებული იყო თითოეულ როუტერზე, მე-2 და მე-3 ეტაპზე ყველაზე დაკავშირებული კვანძი ხელოვნურად გაითიშა და შემდეგ ისევ ჩაირთო. საბოლოო ეტაპზე მოხდა LSA-ის განახლება.

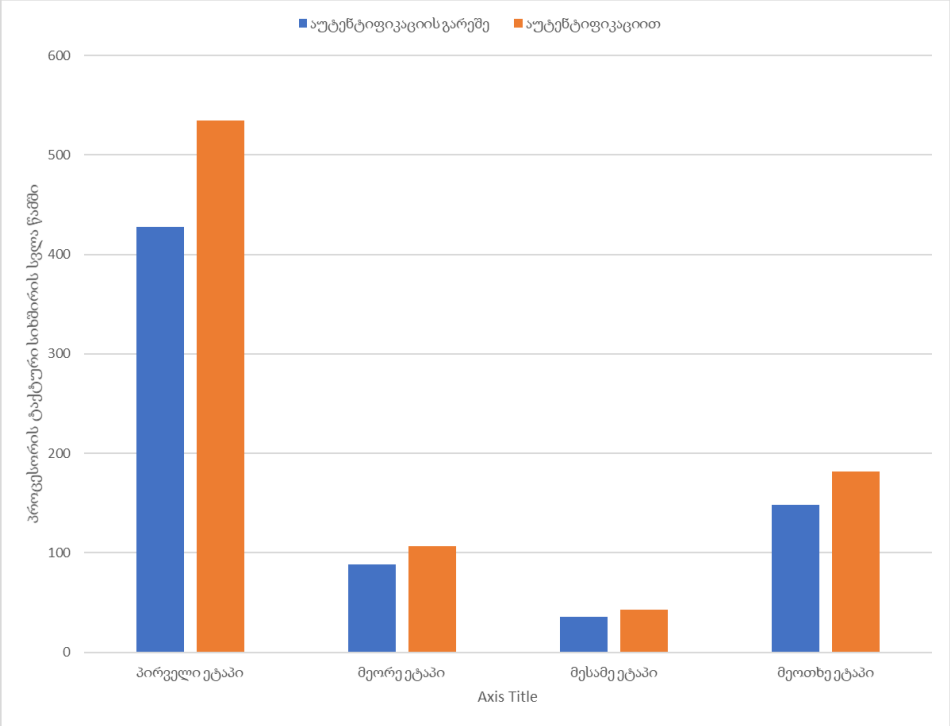
ექსპერიმენტი ჩატარებული იყო სახლის კოპიუტერზე შემდეგი მონაცემებით:

პროცესორი : Intel® Core™ i5-4590 CPU @ 3.30 GHz

ოპერატიული: 8.00 Gb

ვიდეოკარტა: NVIDIA GeForce GTX 660 Ti

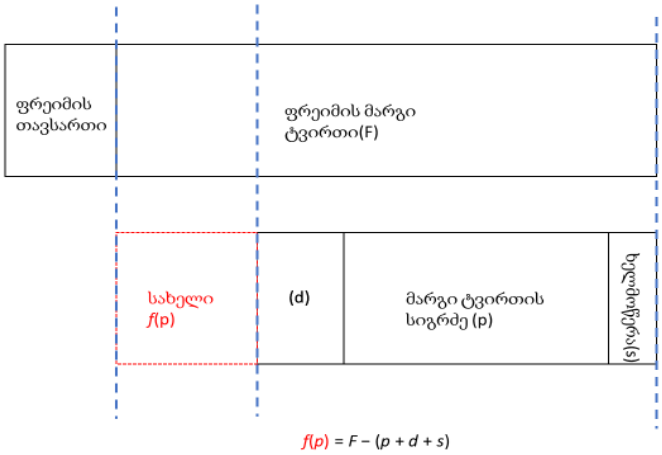
ჩვენ ნდობის მოდელის პირობებშიც კი, რომელიც მოითხოვს გასაღების მრავალდონიან ვერიფიკაციას, NLSR დიდად ვერ ზრდის დატვირთვას. გასაღების აუტენტიფიკაცია დამატებით ამატებს საშუალოდ 20-25%-ს დატვირთვას მხოლოდ ჩვენ 4 განსაზღვრულ ეტაპში. სურათ 4_ზე იხილეთ დატვირთვის შედეგები. ნახ. 29-ზე იხილეთ CPU-ს დატვირთვა NLSR გასაღების აუტენტიფიკაციის არსებობის და არ არსებობის შემთხვევაში.



ნახ. 29. CPU-ს დატვირთვა NLSR გასაღების აუტენტიფიკაციის არსებობის და არ არსებობის შემთხვევაში

გასათვალისწინებელია რომ უზარმაზარი NLSR ტოპოლოგიის შემთხვევაში(300+ მეზობელი) LSA_ების რაოდენობა იქნება გაზრდილი და ქსელის კომუტაციის დაყოვნება შეიძლება გაიზარდოს რამოდენიმე წამით, თუმცა ასეთი ტოპოლოგიის შემქმნა წესით არ უნდა ხდებოდეს და გადაწყვეტილი უნდა იყოს მარშუტების შეჯამებით.

2.5 NDN პაკეტის სახელის ზომის განსაზღვრა და პაკეტის ფრაგმენტაცია.



ნახ. 30. პაკეტის სახელის ზომის განსაზღვრა

ნახ.30-დან ნათლად ჩანს, რომ ფრეიმის მარგი ტვირთის გაზრდით მცირდება კონტენტის სახელის ზომა. იმისათვის რომ ეფექტურად გავმართოთ მცირე ზომის ფრეიმები უნდა შევიმუშაოთ მონაცემთა პაკეტის სწორი მენეჯმენტი, ვინაიდან NDN-ში არ არსებობს წინასწარ განსაზღვრული ზომა და ფორმატი. სწორედ ამიტომ უნდა გვესმოდეს თანაფარდობა სახელის სიგრძისა, მარგი ტვირთის და ველებს შორის მონაცემთა პაკეტში. აქედან გამომდინარე, ჩვენ გვთავაზობთ Named-Payload-Field დაბალანსების ფუნქციას, თითოეული მახასიათებლის გასაკონტროლებლად.

განვიხილოთ შემდეგი მაგალითი:

F - ფრეიმის მარგი ტვირთის ზომა

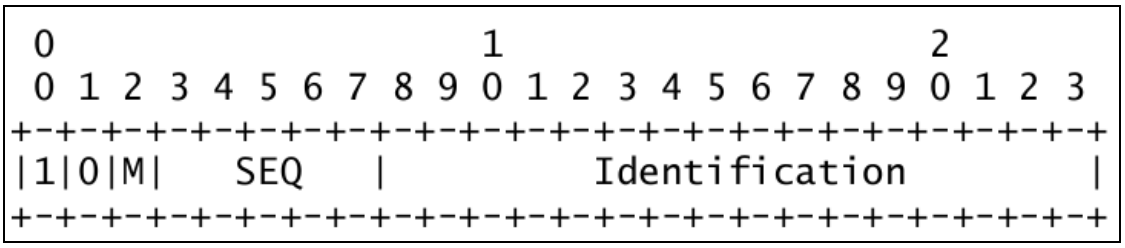
D - მონაცემთა პაკეტის სიგრძე განსაზღვრული ველებით. გათვალისწინებულია პაკეტის ყველა ველი, სახელის და კონტენტის გარდა, იგივეა რაც თავსართი name-ის გარეშე.

P - მონაცემთა პაკეტის მარგი ტვირთის სიგრძე

S - მონაცემთა პაკეტში ხელმოწერის სიგრძე.

f(p) - სახელის დაშვებული ზომა მარგი ტვირთის შესაბამისად განისაზღვრება: f(p) = F – (p + d + s) [1]

აღსაინშნავია, რომ d მახასიათებელი იცვლება დინამიურად. იმისათვის რომ გავიგოთ თანაფარდობა მონაცემთა პაკეტსა და d მახასიათებელს შორის, უნდა გამოვიყვანოთ ფუნქცია, რომელიც მოგვცემს მონაცემთა პაკეტის შეფასების საშუალებას, მონაცემთა პაკეტის ველების გათვალისწინებით. MTU დაბალსიჩქარიან უსადენო ქსელებში საშუალოდ 100 ბაიტის ტოლია, აღსანიშნავია რომ IoT აპლიკაციები არ გადასცემენ დიდი მოცულობის მონაცემებს ერთი პაკეტით, ამიტომ ალბათობა იმისა რომ ხელმოწერილი NDN-ის პაკეთი ჩაეტევა მცირე ზომის MTU-ში არის პატარა, ვინაიდან პაკეტში გამოიყენება Public key კრიფტოგრაფია. ასეთ შემთხვევაში მსუბუქი საარხო დონის ფრაგმენტაციის მექანიზმის გამოყენება იქნება ხელსაყრელი, ის გამოიყენებს 3 ბაიტიან თავსართის ფრაგმენტაციას[16]. ნახ. 31-ზე ნაჩვენებია პაკეტის ფრაგმენტაცია.



ნახ. 31 პაკეტის ფრაგმენტაცია

თუკი თავსართში პირველი ბიტი ერთის ტოლია, მაშინ ზემოთ აღნიშნული მიუთითებს პაკეტის ფრაგმენტაციაზე. არაფრაგმენტირებული პაკეტი იწყება 0-ით. More Fragment(MF) –ამოწმებს პაკეტს ფრაგმენტაციაზე. SEQ (მიმდევრობის რიცხვი) და საიდენტიფიკაციო ველები მიუთითებენ ფრაგმენტების თანმიმდევრობაზე, რომელიც გამოიყენება მიმღების მხარეს პაკეტის ხელახალი აწყობისთვის. [15]

2.6 ჩატარებული ანალიზის ძირითადი არსი და კვლევის გაგრძელების შემდეგი ეტაპი

ჩემს მიერ იყო შესწავლილი და ემულირებული NDN ქსელი MiniNDN პროგრამული უზრუნველყოფის საშუალებით. ქსელის განსახორციელებლად დაგვჭირდა MiniNDN-ის source კოდში გარკვევა, დამატებით ndn-tool-ების დაყენება და Python-ზე სკრიპტის დაწერა, რომელიც უზუნველყოფდა: ახალი დინამიური კვანძის დამატებას ქსელში, კვანძებს შორის მარშრუტების გაწერას როგორც IP ასევე NDN-ის ბაზაზე , და რაც მთავარია დაქეშირებული ინფორმაციის გაზიარებას.

მოყვანილ მაგალითზე ნაჩვენები იყო , ჩემს მიერ დაწერილი სკრიპტის ფუნქციონალის ტესტირება, კერძოდ დაყოვნების შედარება TCP/IP და NDN ქსელს შორის. TCP/IP შემთხვევაში მინიმალური დაყოვნების დრო შეადგენდა 61.0 მწმ. NDN-ის შემთხვევაში კი 22.883 მწმ. ამ მაგალითიდან ნათლად ჩანს, რომ NDN-ის ბაზაზე შემუშავებული ალგორითმების საფუძველზე შესაძლებელია დაყოვნების შემცირება. მეორე ექსპერემენტში, NDN პაკეტის გასაღების აუტენტიფიკაციის არსებობის და არ არსებობის პირობებში, შევაფასეთ CPU დათვირთვა კონკრეტულ პირობებში. გასაღების აუტენტიფიკაცია დამატებით ამატებს საშუალოდ 20-25%-ს CPU დატვირთვას, სამაგიეროდ NDN-ში გამოიყენება დაშიფრვის მექანიზმი SHA-256, რომელიც სადღეისოდ, ყველაზე საიმედო სისტემად ითვლება ; ასევე დავადგინეთ რომ NDN პაკეტის მარგი ტვირთი უკუპროპორციულია კონტენტის სიგრძისა; მაგ.: NDN პაკეტის მარგი ტვირთის გაზრდით კონტენტის სიგრძე მცირდება.

Mini-NDN-ი არ არის სრულყოფილი პროგრამული უზრუნველყოფა. იმიტომ რომ დამუშავების სტადიაშია. მაგ.: როუტერის მეზობლები მხოლოდ სტატიკურები არიან, ამიტომ ჩემი მიზანია მომავალში NLSR-ის დამუშავება და კერძოდ დინამურობის შექმნა რაც ითვალისწინების

კონფიგურაციის ფაილის გაშვების შემდეგ Mini-NDN-ის ტერმინალიდან ახალი მეზობლების დამატებას. ჩემს მიერ დაწერილი ალგორითმებით უნდა განვახორციელო NDN ქსელის და IoT ტექონლოგიის ემულაცია კერძოდ, NDN ქსელში ჩავრთოთ სენსორები და შევადაროთ NDN ქსელის ეფექტურობა არსებულ TCP/IP ქსელს, ასევე მსურს ერთ ინტერესიანი ინფორმაციის ერთი სიხშირით გავრცელება რაც მნიშვნელოვნად დაზოგავს სიხშირულ რესურსს.

თავი III. NDN-ის ინტეგრაცია არსებულ ქსელთან.

3.1 Ipv4-ის და Ethernet frame-ის შედარება

IPv4 პროტოკოლი აღწერილია RFC 791 სპეციფიკაციაში და გამოქვეყნებულია IETF-ის მიერ 1981 წელს. ითვლება ყველაზე ფართოდ გავრცელებულ ქსელურ დონეზე მომუშავე პროტოკოლად. კვლევებზე დაყრდნობით სტატისკიკამ აჩვენა, რომ IPv4-მა გაცილებით ფართო გამოყენება პოვა ვიდრე IPv6-მა. ქვემოთ მოცემულ ნახ.32-ზე ნაჩვენებია IPv4 პაკეტის თავსართის სტრუქტურა[2]

+	ბიტები 0-3	4-7	8-15	16-18	19-31
0	ვერსია	ინტერნეტის თავსართის სიგრძე	სერვისის ტიპი	ჯამური სიგრძე	
32	იდენტიფიკატორი			დროშები	ფრაგმენტის წაწევა
64	TTL		პროტოკოლი	თავსართის ჯამი	
96	შექმნის წყაროს მისამართი				
128	მიმღების მისამართი				
160	პარამეტრები (წებაყოფლობითი)				
160 აწ 192 +	მონაცემები				

ნახ. 32. IPv4 პაკეტის თავსართის სტრუქტურა

IPv4-ის პაკეტში თავსართის პირველი ველი განსაზღვრავს პაკეტის ვერსიას. IPv4-ის შემთხვევაში პაკეტის თავსართის ვერსიის ველი წარმოდგენილია 4 ბიტით.

ინტერნეტის თავსართის სიგრძე - IP პაკეტის თავსართის სიგრძე იზომება “word”-ებში . მაგ.: თუკი თავსართის სიგრძე შედგება 20 ბაიტისგან მაშინ მისი მნიშვნელობა 5-ის ტოლია, რაც ნიშნავს იმას რომ ის შედგება 5

ცალი 32-იანი “word”-ისგან. პაკეტის თავსართის სიგრძე არ არის ფიქსირებული ზომის და მერყეობს 20 ბაიტიდან - 60 ბაიტამდე, ვინაიდან ზომა დამოკიდებულია აგრეთვე მეორეხარისხოვანი პარამეტრების ველის სიგრძეზე.

სერვისის ტიპი - მაგ.: ხომავანი სერვისი, მონაცემთა გადაცემის სერვისი.

ჯამური სიგრძე - პაკეტის სიგრძე განისაზღვრება 16 ბიტიანი მიმდევრობით, რომელიც მოიცავს როგორც თავსართს, ასევე მონაცემებს. ჯამური სიგრძე მერყეობს 20 ბაიტიდან - 65535 ბაიტამდე. თუკი ჯამური სიგრძე გადააჭარბებს მაქსიმალურ ზომას მაშინ პაკეტის დაამუშავება განხორციელდება ფრაგმენტაციის საშუალებით. [2]

იდენტიფიკაცია - 16 ბიტიანი მიმდევრობა, რომელიც გამოიყენება IP პაკეტის ფრაგმენტების იდენტიფიცირებისთვის.

დროშები - 3 ბიტიანი მიმდევრობა, რომელიც დაკავშირებულია ფრაგმენტაციის დაწყების დროსთან.

ფრაგმენტის წაწევა - უზრუნველყოფს პაკეტის ფრაგმენტაციას და აგრეთვე პაკეტის დამუშავებას MTU-ს გათვალისწინებით, რათა შემდეგ მოხდეს ფრეიმში პაკეტის მოთავსება.[16]

TTL - შედგება 8 ბიტისგან და განისაზღვრება როგორც პაკეტის არსებობობის ხანგრძლივობა. ადრე იზომებოდა წამებში, მაგრამ ახლა მოიხსენება ლიტერატურაში როგორც ნახტომების რაოდენობა კვანძებს შორის.

პროტოკოლი - 8 ბიტიანი მიმდევრობა; მაგ.: TCP პროტოკოლი ასოცირებულია 6-თან, ხოლო UDP კი 17-თან. ასევე აქვს მხარდაჭერა ისეთი პროტოკოლების როგორიც არის: ARP, ICMP. რიცხვი განსაზღვრავს პორტის ნომერს.[16]

თავსართის ჯამი - შედგება 16 ბიტისგან, ეს ველი გამოიყენება თავსართში არსებული შეცდომების შემოწმებისთვის.

გადამცემის მისამართი - 32 ბიტიანი მიმდევრობა, ამ ველში ჩაიწერება გამგზავნის IP მისამართი.

მიმღების მისამართი - 32 ბიტიანი მიმდევრობა, ამ ველში ჩაიწერება მიმღების IP მისამართი.

პარამეტრები - წარმოადგენს მეორეხარისხოვან ველს და ფართოდ არ გამოიყენება.

მონაცემები - მარგი ტვირთი;საინფორმაციო ნაკადი.[16]

ქსელის ანალიზატორით დაფიქსირებული IP პაკეტის მაგალითი:

ვერსია	4
თავსართის სიგრძე	5
პრიორიტეტი	0
სერვისის ტიპი	%000
ჯამური სიგრძე	187
იდენტიფიკატორი	22486
ფრაგმენტაციის დროშები	%010
ფრაგმენტაციის წაწევა	0
TTL	60
თავსართის ჯამი	0xd031
გადამცემის IP მისამართი	10.7.1.30
მიმღების IP მისამართი	10.7.1.10
პარამეტრები	none

ნახ. 33. IP პაკეტის სტრუქტურა

პროტოკოლის და პარამეტრების ცვლილებით ტექნიკურად შესაძლოა NDN პაკეტის ჰედერის ნაწილი წაინაცვლოს IPv4-ის პაკეტის

ჰედერში. “Options” ველი ხშირად გამოყენებულია მარშრუტიზაციის სხვადასხვა პროტოკოლში და როგორც უკვე ადრე ისტორიამ აჩვენა პროგრამული უზრუნველყოფის მარშრუტიზატორებზე ახალი პროტოკოლების მხარდაჭერა კეთდება პროგრამული უზრუნველყოფის განახლებით. შემზღუდავი ფაქტორი შეიძლება იყოს მარტო მარშრუტიზატორის ფიზიკური რესურსი. მთავარი მიზეზი რატომაც მოვიყვანეთ ალტერნატიული მაგალითი თუ როგორ შევათავსოთ IPv4 NDN-თან არის ის, რომ ფიზიკურად სუსტი მოწყობილობების შემთხვევაში NDN-ის დეტალური დამუშავება ვერ მოხდება და საჭირო იქნება მონაცემების სამარიზაცია ე.წ. მარშრუტების შეჯამება და ჩვენს მიერ შემოთავაზებული ველები ამისთვის სრულიად საკმარისია.

```
struct iphdr {
#if __BYTE_ORDER == __LITTLE_ENDIAN
   unsigned int ihl: 4;
   unsigned int version: 4;
#elif __BYTE_ORDER == __BIG_ENDIAN
   unsigned int version: 4;
   unsigned int ihl: 4;
#else
# error "Please fix <bits / endian.h>"
#endif
   u_int8_t tos;
   u_int16_t tot_len;
   u_int16_t id;
   u_int16_t frag_off;
```

```
    u_int8_t ttl;
    u_int8_t protocol;
    u_int16_t check;
    u_int32_t saddr;
    u_int32_t daddr;
    u_int8_t ndn_test[40];  // NDN-თვის გამოყოფილი 40 ბაიტი
}; [17]
```

40 ბაიტზე მეტის გამოყოფა IPv4 სტანდარტის გათვალისწინებით ვერ ხდება, ამიტომ ამ 40 ბაიტში უნდა მოვათავსოთ ყველაზე საჭირო და შეჯამებული NDN ინფორმაცია.

საარხო დონე უზრუნველყოფს ბიტების გაერთიანებას ბაიტებში და ბაიტების გაერთიანებას ფრეიმებში. OSI მოდელის თანახმად ქსელური დონიდან მოსული პაკეტების დამუშავება და შემდგომ მათი ფრეიმებში ენკაფსულაცია ხორციელდება საარხო დონეზე. ნახ. 34-ზე ნაჩვენებია Ethernet-ის ფრეიმი.

პრეამბულა	ფრეიმის დასაწყისი	მიმღების მისამართი	წყარო	ტიპი	მონაცემები	ფრეიმის შემოწმების მიმდევრობა
7 ბაიტი	1 ბაიტი	6 ბაიტი	6 ბაიტი	2 ბაიტი	46-1500 ბაიტი	4 ბაიტი

ნახ. 34. Ethernet-ის ფრეიმი

პრეამბულა: 1 და 0 თანმიმდევრობის შაბლონი, რომელიც გამოიყენება ethernet კადრის სინქრონიზაციისთვის.

ფრეიმის დასაწყისი - პრეამბულა შედგება 7 ოქტეტისგან და წარმოადგენს. სინქრონიზაციის ოქტეტს. SFD – 10101011, ამ მიმდევრობით მიმღები მოწყობილობა ხვდება თუ საიდან იწყება მონაცემთა ნაკადი.

მიმღები მოწყობილობის მისამართი - მიმღები მოწყობილობის 48 ბიტიანი MAC მისამართი.[17]

გადამცემი მოწყობილობის მისამართი – გადამცემი მოწყობილობის 48 ბიტიანი MAC მისამართი.

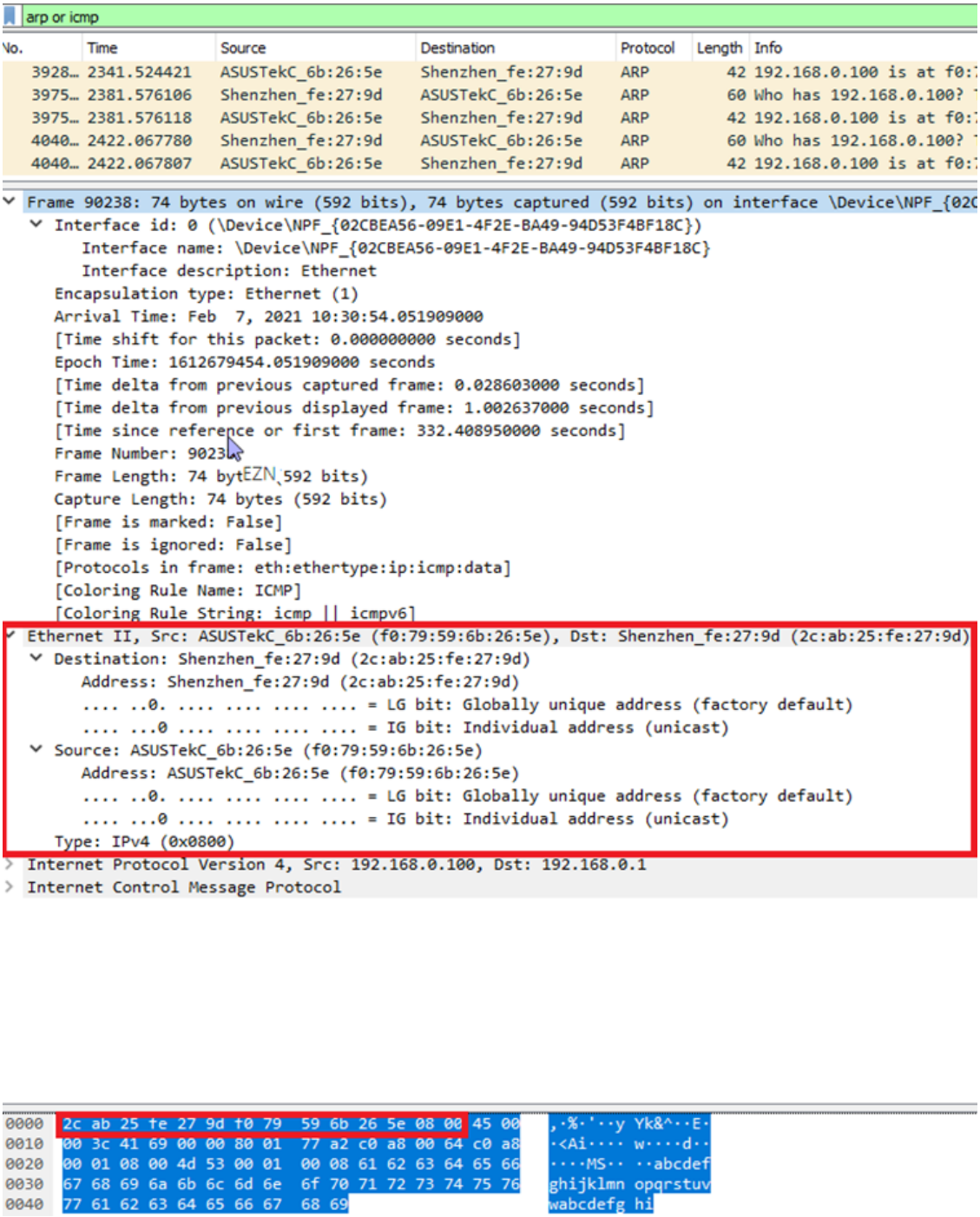
სიგრძე - ფრეიმის ზომა არ უნდა აღემატებოდეს 2 ბაიტს.

მონაცემები - მარგი ტვირთი, ზომა მერყეობს 46 ბაიტიდან – 1500 ბაიტამდე.

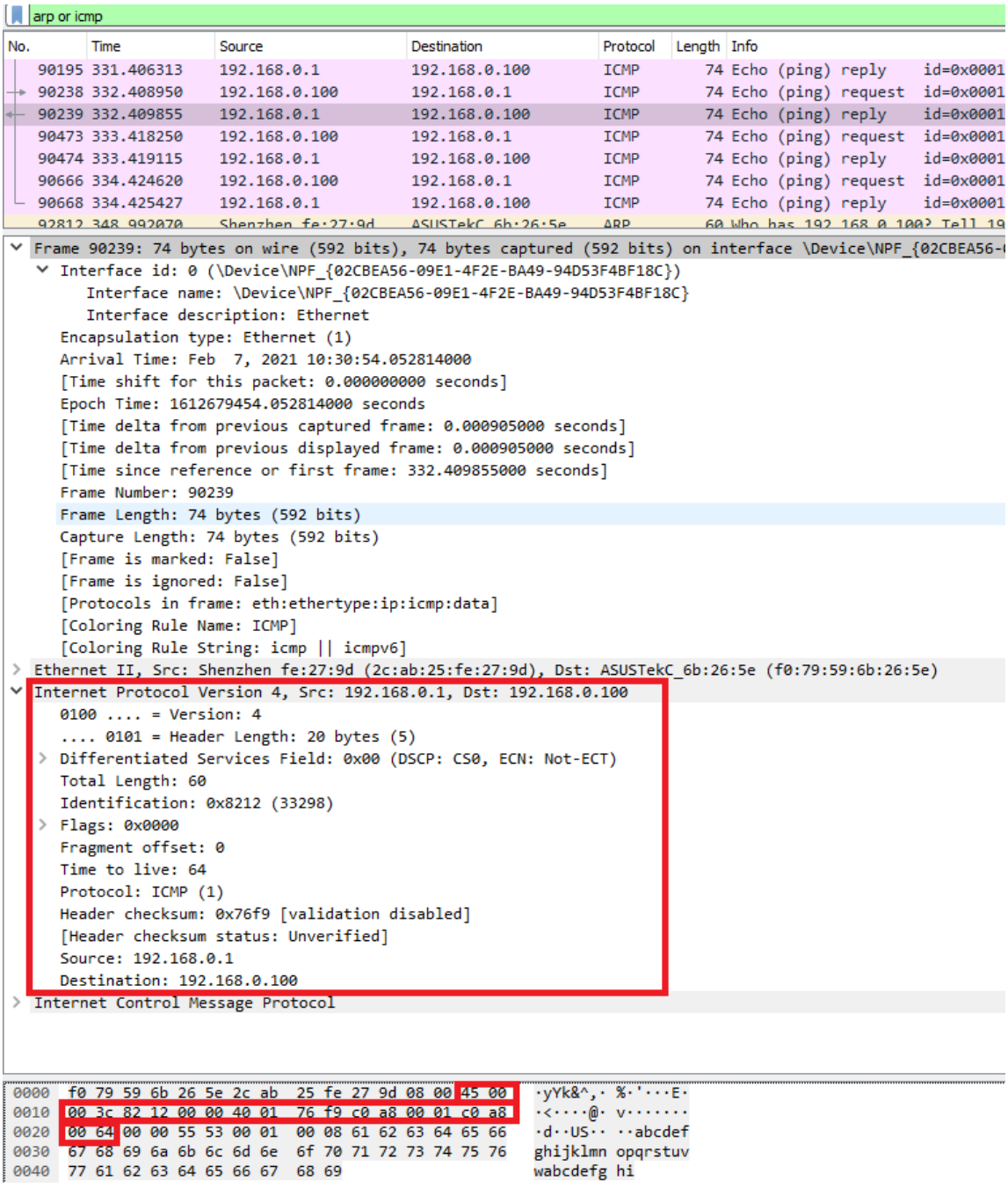
ფრეიმის შემოწმების მიმდევრობა – ამ ველში ინახება CRC მნიშვნელობა. მიმღებ მოწყობილობაზე მოსული ფრეიმის CRC მნიშვნელობა უნდა ემთხვეოდეს გადამცემი მოწყობილობის CRC-ს, წინააღმდეგ შემთხვევაში ფრეიმი გაუქმდება. სურათზე წითლად მონიშნულია ფრეიმი, თუმცა არ ჩანს შემოწმების ბიტები ე.წ. CRC, რადგან იფილტრება ინტერფეისის მოწყობილობაზე.[17]

ნახ. 35-ზე და ნახ. 36-ზე მოყვანილია ფრეიმისა და IPv4 სექციები. მოგეხსენებათ, რომ IPv4-ის პაკეტი იდება ფრეიმის შიგნით, ამ პროცესს ეწოდება ენკაფსულაცია. მსგავსი პრინციპით შეგვიძლია მოვახდინოთ NDN პაკეტის გადაცემა ქსელში არსებული ფრეიმების საშუალებით IPv4-ის მაგივრად და ასევე შეგვიძლია ჩავდოთ NDN-ის პაკეტი არსებულ IPv4 პაკეტში, მაგრამ ეს გამოიწვევს დამატებით დატვირთვას, დაყოვნებას და ა.შ. თუმცა პირდაპირ რომ ჩავდოთ ფრეიმში ზოგმა მოწყობილობამ შეიძლება ვერ წაიკითხოს პაკეტები, ვინაიდან მათ არ ექნებათ ამის მხარდაჭერა და შედეგად მოხდება პაკეტების გადაგდება. კარგი იქნებოდა ჰიბრიდული ვარიანტის გამოყენება, რაც გულისხმობს IPv4 პაკეტის, კერძოდ “Option” ველის ცვლილებით ჩვენი NDN პაკეტის ჩამატებას და შემდგომ გაგზავნას

ქსელში. NDN-ის პაკეტის გაგზავნა არსებულ ქელში პირდაპირი გზით ვერ განხორციელდება, ამისთვის საჭიროა NDN-ის პაკეტის ენკაფსულაცია.



ნახ. 35. ფრეიმის სექცია



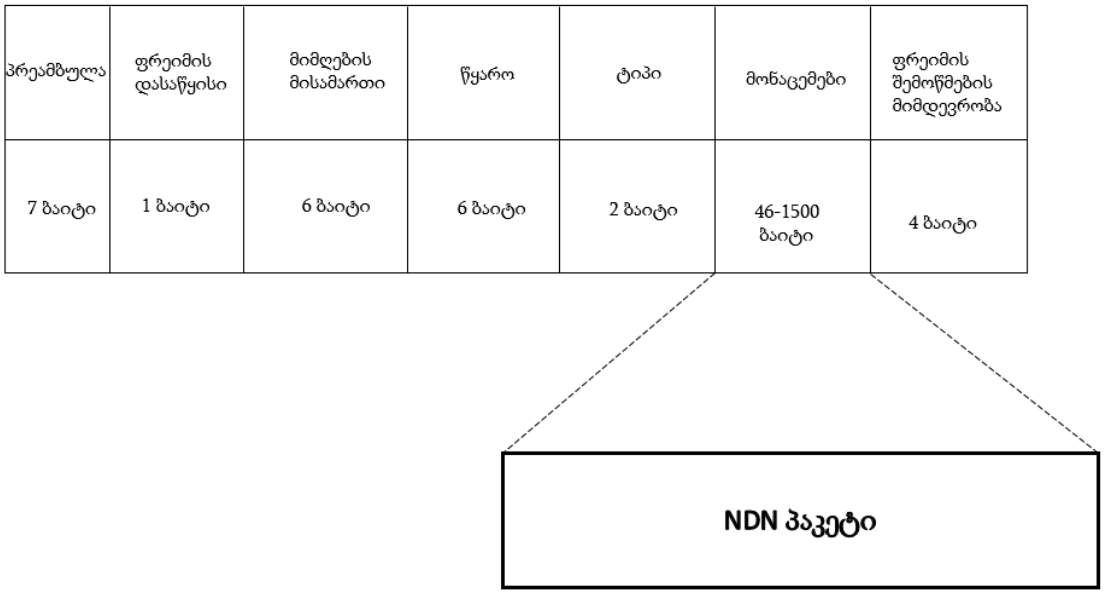
ნახ. 36. IPv4 სექცია.

3.2 Ipv4-ის და NDN პაკეტების შედარება

NDN პაკეტი შეგვიძლია გადავცეთ ქსელში ეზერნეთის საშუალებით ამისთვის უნდა ჩავამატოთ NDN პაკეტი ფრეიმში, კერძოდ კი მონაცემთა ველში. მაგრამ შეზღუდული ვართ ზომაში იმიტომ რომ NDN პაკეტის ზომა IPv4 პაკეტისგან განსხვავებით არის განუსაზღვრელი ზომის. როგორც IPv4

პაკეტი სრულად ვერ ეტევა ფრეიმის 1500 ბაიტიან საინფორმაციო ველში ასევე მსგავსი შემთხვევაა NDN პაკეტების შემთხვევაში. აქ საჭირო იქნება NDN პაკეტის ფრაგმენტაცია. აღსანიშნავია ის, რომ IPv4-ის ფრაგმენტაციის შემთხვევაში ფრაგმენტირების ინფორმაცია ჩადებული იყო IPv4-ის ჰედერშივე, კერძოდ კი დროშებსა და ფრაგმენტირების წაწევის ველებში. NDN პაკეტის ენკაფსულაციის შემთხვევაში ჩვენ გვთავაზობთ ახალი ველების დამატებას. IPv4 შემთხვევაში პაკეტის ზომისთვის განსაზღვრული იყო 16 ბიტი. თითო მნიშვნელობა შეესაბამებოდა ერთ ბაიტს და შესაბამისად აქ პაკეტის მაქსიმალური ზომა შეზღუდული იყო 216 ბაიტით. NDN-თვის 64 ბაიტის გამოყოფის შემთხვევაში ჩვენ გვექნება 264 ბაიტი, რაც შეადგენს 18,446,744,073,709,551,616 ბაიტს. ასეთი დიდი რიცხვი უნდა ეყოს სადღეისოდ ყველაზე დიდ ფაილებსაც კი. საქმე იმაშია, რომ მიზანშეწონილია ფაილის ერთ პაკეტში დატევა. ეს საჭიროა იმისთვის, რომ ინტერესთა პაკეტის გაგზავნის შემთხვევაში მოთხოვნა მოხდეს მთლიან ფაილზე და არა მის ნაწილზე. ალტერნატიული ვარიანტი იქნებოდა პოსტამბულის გაკეთება, თუმცა ზოგიერთი ფაილის შემთხვევაში არის საშიშროება, რომ მოხდეს სანფორმაციო ნაწილის პოსტამბულასთან არევა. ამის გადაჭრის გზა იქნებოდა ზემოთ აღნიშნული მონაკვეთის დამუშავება, მაგრამ გართულდებოდა იმისდა მიხედვით თუ რამხელაა ფაილი, ამიტომ ამის რეალაზიცია ამ ეტაპზე მიმაჩნია არაპრაქტიკულად. ქვემოთ მოცემულ ნახ.37 ზე ნაჩვენებია თუ სად იქნება მოთავსებული NDN-ის პაკეტი ფრეიმში.

მინდა შემოგთავაზოთ NDN-ისა და IPv4-ის ჰიბრიდული ვარიანტი. NDN პაკეტის დამუშავება შეგვიძლია გავაკეთოთ ქსელურ დონეზე IPv4-ის მოწყობილობების მონაწილებით. IPv4-ის ქსელის გადაწყობა სრულ NDN ქსელში იქნება საკმაოდ შრომატევადი და დაკავშირებული დიდ ხარჯებთან, ამიტომ ვფიქრობ მიზანშეწონილი იქნებოდა ნაწილობრივი მიგრაცია.



ნახ. 37. NDN პაკეტის ფრეიმში მოთავსების სქემა

თეორიაში მთელი ქსელის გადაწყობა შესაძლებელია შესაბამისი პროგრამული უზრუნველყოფის დააფგრეიდებით. სამწუხაროდ, აქ შემზღუდავი ფაქტორი დარჩება ქსელში არსებული მოწყობილობების ფიზიკური შესაძლებლობები. დღევანდელ მოწყობილობებს არ აქვთ შენახვის დისკები ან არის მცირე ზომის. შენახვის თავისუფალი მოცულობა არ არის საკმარისი დიდი ფაილების შესანახად, თუმცა საკმარისი უნდა იყოს მცირე ინფორმაციებისთვის, როგორიც არის მაგალითად: გზის დატვირთულობა, კონკრეტულ უბანში ტენიანობა ან ჰაერის დაბინძურება. სადღეისოდ IoT-ის ტენდენციით ქალაქში შეგვიძლია დავაყენოთ მრავალი სენსორი, რომელიც არ მოიცავს ინფორმაციის სიდიდეს. ასეთი ინფორმაციის შენახვა საკმაოდ რეალური არსებული Ipv4 მოწყობილობების საშუალებით. მოწყობილობების განახლება უნდა მოხდეს შესაბამისი კომპანიების მიერ. გასათვალისწინებელია სტანდარტიზაცია და თავსებადობა მომავალ ვერსიებთან. არის შემთხვევები სადაც მოწყობილობებზე პროგრამული უზრუნველყოფის ტექნიკური მხარდაჭერა მიტოვებულია კომპანიების მიერ. საბედნიეროდ, სხვადასხვა მწარმოებლის მოწყობილობებზე ხშირად მოიძებნება ერთი და იგივე ჩიპსეტები, რაც ხელს

შეუწყობს არაორიგინალი, მოდიფიცირებული პროგრამული სისტემების დაყენებას. ასეთი მაგალითები უკვე დიდი ხანია რაც არის და იმის გათვალისწინებით, რომ NDN პლატფორმა ვითარდება ღია წვდომის სისტემებზე, როგორიც არის Ubuntu, NDN-ის პორტირება ისეთი მოწყობილობებისთვის არ უნდა შეადგინოს ტექნიკურ სირთულეებს. მოწყობილობებზე, რომლებზე არ არაის საკმარისი თავისუფალი სივრცე შესაბამისად ექნებათ მარტო მარშრუტიზაციის ფუნქცია. IPv4-ის ენკაფსულაციასგან განსხვავებით, აქ მოწყობილობა შეხედავს მხოლოდ ჰედერის მცირე ნაწილს, რადგან კონტენტ სტორში ჩახედვის აზრი მისთვის არ გააჩნია. NDN ველის ნაწილობრივი გადატანა IPv4-ში მოხდება IPv4-ის პაკეტის ოფციის ველებით. ეს ველი შეზღუდულია 40 ბაიტით, თუმცა ამ ველის შემდეგ მოდის IPv4-ის ფეილოუდი, სადაც ასევე თეორიაში შესაძლოა NDN ჰედერის მონაცემების გადატანა. მაგალითად, ოფციებში შეგვიძლია გამოვუყოთ 4 ბაიტი ოფციების ფეილოუდში გაგრძელებისთვის. შესაბამისად, ოფციების დამთავრების შემდეგ ჩვენ გავიმეორებთ ორიგინალურ NDN პაკეტის ფეილოუდს. შესაძლოა NDN-ის ენკაფსულაცია IPv4-ის ფეილოუდშივე. ქსელურ დონეზე IPv4-ის სტეკზე იქნება აგებული. ამიტომ ქსელში არსებული მოწყობილობები შეძლებენ NDN პაკეტის თავსართის წაკითხვას ანუ წაიკითხავენ ყველაზე მნიშვნელოვან ინფორმაციას და შემდგომ გადასცემენ ამ პაკეტებს სხვა კვანძებზე.

შესაძლებელია NDN პაკეტის გადაცემა შემდეგი გზით: NDN პაკეტის ნაწილის IPv4 პაკეტის მონაცემებში და ნაწილის IPv4 პაკეტის პარამეტრებში ჩასმა. ასეთ შემთხვევაში ის დაიკავებს შემდეგ ბიტებს იხილეთ ნახ.30 წითელი მართკუთხედის მერე ბიტების მიმდევრობა მიუთითებს NDN პაკეტის სიგრძეზე. ასევე შესაძლებელია NDN პაკეტის ჩასმა პირდაპირ მონაცემებში, პარამეტრების გარეშე. ამ მეთოდს გააჩნია დიდი მინუსი, კერძოდ ქსელში არსებული მოწყობილობები არ იციან NDN-ის პაკეტის სტრუქტურა და ამიტომ ვერ წაიკითხავენ. ასევე საჭირო იქნება

ახალი მარშრუტიზაციის შექმნა წყაროსა და მიმღებს შორის, ის არ იქნება დინამიური.

თეორიულად შესაძლებელია ჰიბრიდული ვარიანტის გამოყენება ანუ ნდნ-ის მიგრაციით არსებულ ქსელში იმ პირობით რომ სხვა მოწყობილოებს ექნებათ NDN-ის მხარდაჭერა მაშინ პირდაპირ frame-ის მონაცემებში ჩავსვავთ NDN-ს პაკეტებს და ეზერნეტის საშუალებით გადავცემთ მათ ქსელში.

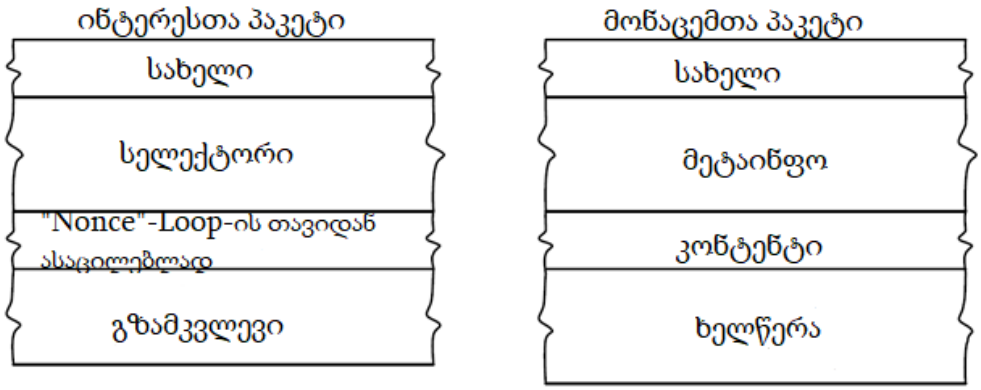
NDN პაკეტის ჩატევა IPv4-ში შესაძლებელი იქნება ფრაგმენტაციის გზით. რაც ხდება ფლაგებსა და ფრაგმენტაციის ოფსეტში.

3.3 NDN პაკეტების სტრუქტურა

NDN ქსელში არსებობს პაკეტის 2 ტიპი:

- ინტერესთა პაკეტი;
- მონაცემთა პაკეტი;

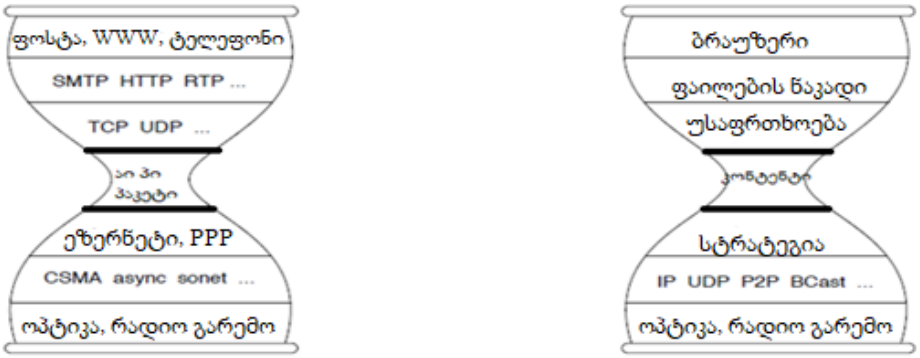
ნახ. 38-ზე მოყვანილია NDN-ში არსებული პაკეტების სტრუქტურა.



ნახ. 38. NDN-ში არსებული პაკეტების სტრუქტურა

- ინტერესი: სასურველი ინფორმაციის დასახელება მომხმარებლის მიერ ჩაიწერება ინტერესთა პაკეტში და შემდგომ იგზავნება ქსელში.
- მონაცემები: მარგი ტვირთის ინფორმაცია, ანუ მწარმოებლის მიერ დაბრუნებული საინფორმაციო პაკეტი, რომელიც მოიცავს: მოთხოვნის დასახელებას და შესაბისი ინფორმაციის კონტენტს. მწარმოებლის მიერ სპეციალური გასაღებით ზემოთაღნიშნული პაკეტი იქნება დაშიფრული.

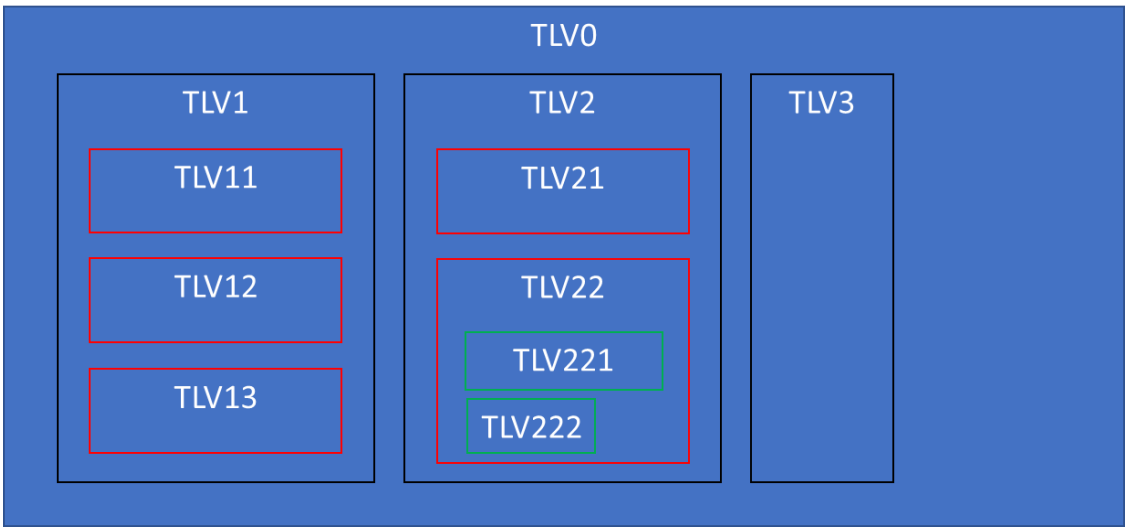
ნახ. 39-ზე მოყვანილია საქსელო შრის სპეციფიკა.



ნახ. 39. საქსელო შრის სპეციფიკა

როგორც მე-5 ნახაზზე ჩანს, ძირითადი განსხვავება თანამედროვე ინტერნეტისა და NDN ქსელს შორის მკვეთრად ასახულია OSI-ს მოდელის ქსელური შრის სტრუქტურაში, კერძოდ TCP/IP შემთხვევაში ქსელურ შრეზე გამოიყენება აი პი პაკეტები, NDN-ის შემთხვევაში კი ე.წ. კონტენტი.

NDN-ს არ აქვს ფიქსირებული თავსართი. მას გააჩნია სტრუქტურა, რომელსაც ეძახიან TLV (Type Length Value). NDN პაკეტი შედგება ბევრი TLV-ებისგან, იხილეთ ნახ 40. ფიქსირებული თავსართების არ არსებობა ხელს უწყობს პატარა ზომის პაკეტების ეფექტურად დამუშავებაში. [5]



ნახ. 40. TLV-ის სტრუქტურა

ჰედერში შესაძლებელია ახალი ველების ჩამატება. შეგვიძლია IoT პაკეტები დავთაგოთ სპეციალური TLV-ით , რომლის საშუალებითაც მოხდება FreshnessPeriod-ის TLV-ის განსაზღვრა. FreshnessPeriod არის ცვალებადი. მოგეხსენებათ, რომ NDN-ში ორი სახის პაკეტი არსებობს ინტერესთა და მონაცემთა პაკეტები.

ინტერესთა პაკეტები შედგება:

Interest = INTEREST-TYPE TLV-LENGTH

Name

[CanBePrefix]

[MustBeFresh]

[ForwardingHint]

[Nonce]

[InterestLifetime]

[HopLimit]

[ApplicationParameters [InterestSignature]]

Name - აღსანიშნაცია, რომ ინტერესთა პაკეტში ელემენტი სახელწოდებით Name არის ერთადერთი აუცლებელი ელემენტი.[5]

NDN Name არის იერარქიული სახელწოდება NDN კონტენტის მოსაძიებლად, რომელიც მოიცავს სახელწოდებების მიმდევრობის კომპონენტებს. ჩვენ ვიყენებთ 2 დონიან TLV სახელწოდების წარმოსადგენად. NAME TYPE გარე TLV- ში მიუთითებს Name-ზე, შიდა TLV უნდა იყოს NameComponent ელემენტები, მაგალითად: [5]

Name = NAME-TYPE TLV-LENGTH *NameComponent

NameComponent = GenericNameComponent /

ImplicitSha256DigestComponent /

ParametersSha256DigestComponent /

OtherTypeComponent

GenericNameComponent = GENERIC-NAME-COMPONENT-TYPE TLV-LENGTH *OCTET

ImplicitSha256DigestComponent = IMPLICIT-SHA256-DIGEST-COMPONENT-TYPE

TLV-LENGTH ; == 32

32OCTET

ParametersSha256DigestComponent = PARAMETERS-SHA256-DIGEST-COMPONENT-TYPE

TLV-LENGTH ; == 32

32OCTET

OtherTypeComponent = OTHER-TYPE-COMPONENT-TYPE TLV-LENGTH *OCTET

OtherTypeComponent TLV-TYPE კომპონენტი 1-65535 დიაპაზონში.

GenericNameComponent

ImplicitSha256DigestComponent

ParametersSha256DigestComponent

CanBePrefix = CAN-BE-PREFIX-TYPE[5]

TLV-LENGTH ; == 0

თუკი გამოიყენება ეს ველი მაშინ, Name ელემენტი ინტერესთა პაკეტში პრეფიქსის როლს ასრულებს, და ასახავს დატა პაკეტის ზუსტ ან სრულ სახელწოდებას. ზემოთ აღნიშნული ველის არ არსებობის შემთხვევაში Name ელემენტი ასახავს მონაცემთა პაკეტის ზუსტ ან სრულ სახელოწოდებას.

MustBeFresh

MustBeFresh = MUST-BE-FRESH-TYPE

TLV-LENGTH ; == 0

MustBeFresh-ის ელემენტი განსაზღვრავს კონტენტის აქტუალურობას.

ForwardingHint = FORWARDING-HINT-TYPE TLV-LENGTH 1*Delegation

ForwardingHint ელემენტი მოიცავს დელეგირებული სახელოწოდებების სიას. თითოეული დელეგირება მოიაზრებს იმას რომ მოთხოვნილი მონაცემთა პაკეტი შესაძლოა მიღებულ იქნას ინტერესთა პაკეტის ქსელში დელეგირების გადაცემის გზით.

Nonce განისაზღვრება:

Nonce = NONCE-TYPE

TLV-LENGTH ; == 4

4OCTET

Nonce მოიცავს შემთხვევით დაგენერირებულ 4 ბაიტიან ოქტეტის მქონე სთრინგის ტიპის მნიშვნელობას. ინტერესთა პაკეტის იდენტიფიცირება ხდება Name და Nonce კომბინაციის წაკითხვით.

InterestLifetime = INTEREST-LIFETIME-TYPE TLV-LENGTH NonNegativeInteger

InterestLifetime - განსაზღვრავს პაკეტის არსებობის დროს.

HopLimit განსაზღვრავს ჰოპების რაოდენობას ინტერესთა პაკეტის ქსელში გადაცემის დროს.

HopLimit = HOP-LIMIT-TYPE

TLV-LENGTH ; == 1

OCTET

ApplicationParameters = APPLICATION-PARAMETERS-TYPE TLV-LENGTH *OCTET

Nonce - საჭიროა მაშინ როდესაც ინტერესთა პაკეტი იგზავნება ქსელში კვანძებს შორის.

CanBePrefix, MustBeFresh, InterestLifetime, ForwardingHint არასავალდებულო ელემენტებია.

მონაცემთა პაკეტი განისაზღვრება:

Data = DATA-TYPE TLV-LENGTH

Name

[MetaInfo]

[Content]

DataSignature

მონაცემთა პაკეტი წარმოადგენს ორობითი კოდის მიმდევრობას თანდართულ სახელწოდებასთან ერთად და ასევე დამატებით არა აუცილებელ ველებს: Metainfo, Data Signature.

MetaInfo = META-INFO-TYPE TLV-LENGTH

[ContentType]

[FreshnessPeriod]

[FinalBlockId]

კონტენტის ტიპი განისაზღვრება:

ContentType = CONTENT-TYPE-TYPE TLV-LENGTH NonNegativeInteger

FreshnessPeriod: განსაზღვრავს ინფორმაციის აქტუალობას, გარკვეული პერიოდის შემდეგ ეს ინფორმაცია იშლება კვანძებიდან.

FreshnessPeriod = FRESHNESS-PERIOD-TYPE TLV-LENGTH NonNegativeInteger

FinalBlockId (არა აუცილებელი ველი) - გამოიყენება სინქრონიზაციისთვის.

FinalBlockId = FINAL-BLOCK-ID-TYPE TLV-LENGTH NameComponent

ძირითადი ინფორმაცია იგზავნება Content-ის ველში. ამ ველს აქვს შემდეგი სახე:[5]

Content = CONTENT-TYPE TLV-LENGTH *OCTET

IPv4-ის შემთხვევაში იდენტური ინფორმაცია გადის მრავალგზას და შესაძლოა გამოიწვიოს დაყოვნებები ასევე გარკვეულ უბნებზე გადატვირთვები, მრავალჯერადი მოთხოვნის შედეგად იტვირთება სენსორები რის შედეგადაც ბატარეა ჯდება უფრო სწრაფად. აღსანიშნაცია ის რომ NDN-ის ქსელის გამოყენებამ შესაძლოა ამოიღოს Cloud აუცილებლობა. გვთავაზობთ არსებული ქსელის მოდიფიცირებულ მოდელს ე.წ. NDN-ს, რომელიც IPv4-გან განსხვავებით იდენტური

ინფორმაციის მოსაპოვებლად მხოლოდ ერთხელ გადის მთლიან მარშრუტს. ინფორმაციის მოპოვების შემდეგ კი ამ ინფორმაციას შეინახავს ყველა გავლილ მარშრუტიზატორზე. შედეგად დატვირთვა გარკვეულ უბნებზე შემცირდება, ასევე შემცირდება დაყოვნების დრო, ვინაიდან მომხმარებელი მიიღებს სასურველს ინფორმაციას უახლოესი ქსელის მოწყობილობიდან რომელზეც იქნება მოთხოვნილი ინფორმაცია. NDN ქსელის დანერგვას აქვს თავისი პრობლემები. იქიდან გამომდინარე რომ ეს არის ახალი სისტემა ის არ იქნება თავსებადი არსებულ როუტერებთან. NDN-ის ინტეგრაცია არსებულ ქსელთან შესაძლებელია რამდენიმე გზით:

არსებული მარშრუტიზატორების დააფგრეიდებით.

ფრეიმში NDN პაკეტის ჩადებით.

სადღეისოდ მიდის data-centric ტიპის მონაცემების ტენდენცია. ფიქსირებული მონაცემები იგზავნება. ერთი და იგივე მონაცემების მრავალჯერადი წამოღება იწვევს ქსელის უბნების დატვირთვას. ანუ ერთი და იგივე მონაცემები NDN ტიპის ქსელის გამოყენებით ინფორმაციის განმეორებითი გაგზავნის აუცილებლობა ქრება. ქსელში ჩნდება მრავალი სენსორი რომლის ინფორმაციის წამოღება ხდება მრავალჯერ, მაგრამ სენსორების ინფორმაცია ახლდება დიდი პერიოდებით, მაგალითად 5 წუთში ერთხელ. NDN ქსელი ამცირებს ქსელის უბნებზე დატვირთვას, ამცირებს დაყოვნებას, განტვირთვას სენსორებს. და ზრდის წარმადუნარიანობას.

3.4 მარშრუტიზაციის არსებულ აპარატურაზე NDN-ის ინტეგრაცია.

show version ბრძანებით შევამოწმოთ IOS_ის ვერსია. ტერმინალით გამოტანილი გამოსახულება იხილეთ ნახ. 41-ზე აქ ჩვენ ვხედავთ რომ Virtual XE Software მოისცავს X86_64_Linux_IOSD-UNIVERSALK9-M

ოპერატიულ იმიჯს, რომლის საშვალებითაც მოვახდეთ NDN პლათმორმის დამატებას.

როუტერზე enable ბრძანებით გადავდივართ პრივილეგირებულ რეჟიმში. იმისთვის რომ გადავარქვათ როუტერს სახელი, პირველ რიგში ჩავწეროთ configure terminal და შემდეგ host ბრძანებით გადავარქვათ სახელი. მაგ.:host GTU_NDN. ტერმინალზე გამოტანილი გამოსახულება იხილეთ ნახ. 42-ზე.

```
Router>

Router>

Router>sh ver

Cisco IOS XE Software, Version 16.06.01

Cisco IOS Software [Everest], Virtual XE Software (X86_64_LINUX_IOSD-UNIVERSALK9
-M), Version 16.6.1, RELEASE SOFTWARE (fC2)

Technical Support: http://www.cisco.com/techsupport

Copyright (c) 1986-2017 by Cisco Systems, Inc.

Compiled Sat 22-Jul-17 05:51 by mcpre

Cisco IOS-XE software, Copyright (c) 2005-2017 by cisco Systems, Inc.
All rights reserved. Certain components of Cisco IOS-XE software are
licensed under the GNU General Public License ("GPL") Version 2.0. The
software code licensed under GPL Version 2.0 is free software that comes
with ABSOLUTELY NO WARRANTY. You can redistribute and/or modify such
GPL code under the terms of GPL Version 2.0. For more details, see the
documentation or "License Notice" file accompanying the IOS-XE software,
or the applicable URL provided on the flyer accompanying the IOS-XE
software.
```

ნახ. 41. IOS XE ვერსიის ინფორმაცია

```
License Level: ax

License Type: Default. No valid license found.

Next reload license Level: ax

cisco CSR1000V (VXE) processor (revision VXE) with 2191143K/3075K bytes of memor
y.

Processor board ID 94MW3AOOODG

4 Gigabit Ethernet interfaces

32768K bytes of non-volatile configuration memory.

3984708K bytes of physical memory.

16162815K bytes of virtual hard disk at bootflash:.

0K bytes of WebUI ODM Files at webui:.

Configuration register is 0x2102

Router>enable
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#host GTU_NDN
GTU_NDN(config)#
```

ნახ. 42. სახელის გადარქმევის პროცედურა

გამავალ მარშრუტიზატორზე შევამოწმოთ არსებობს თუ არა მარშრუტი GTU_NDN-ის როუტერამდე. როუტერზე sh ip route ბრძანებით მოწმდება გამავალი მარშრუტიზატორის ინტერფეისებზე გაწერილი მარშრუტები. ტერმინალზე გამოტანილი მარშრუტები იხილეთ ნახ. 42-ზე.

```
GTU_GATEWAY#sh ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
a - application route
+ - replicated route, % - next hop override, p - overrides from PfR
Gateway of last resort is 192.168.1.249 to network 0.0.0.0

S* 0.0.0.0/0 [254/9] Via 192.168.1.249
19.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C 10.1.1.0/24 is directly connected, GigabitEthernet0/0
L 10.1.1.254/32 is directly connected, GigabitEthernet0/0
192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks
C 192.168.1.0/24 is directly connected, GigabitEthernet0/1
L fl 192.168.1.104/32 is directly connected, GigabitEthernet0/1

GTU_GATEWAY#
```

ნახ. 43. გამავალი მარშრუტიზატორის მარშრუტები

GTU_NDN-ის როუტერზე int g1 ბრძანებით შევდივართ გიგაბიტ ეზერნეტის g1 პორტზე და შემდეგ ვამატებთ მისამართს ip address 10.1.1.1 255.255.255.0.

no shut ბრძანებით ხდება ლინკის აწევა down-დან up-ში. end ბრძანებით გამოვდივართ configure terminal-იდან თავდაპირველ მენიუში. ნახ.44-ზე ნაჩვენებია ინტერფეისზე დამატებული IP მისამართი.

Ping 10.1.1.254 ბრძანებით დავპინგოთ ზემოთ აღნიშნული მისამართი. ip int brief-ით შევამოწმოთ არსებული როუტერის ინტერფეისებზე: GigabitEthernet1, GigabitEthernet2, GigabitEthernet3, GigabitEthernet4 გაწერილი მარშრუტები. Unassigned ველი მიუთითებს იმაზე რომ მარშრუტი არ არსებობს.

ნახ.45-ზე ნაჩვენებია GigabitEthernet1-ის ინტერფეისის ip მისამართის ჩამატება ვირტუალურ ჯგუფში.

```
GTU_NDN(config)#int g1
GTU_NDN(config-if)#ip add
GTU_NDN(config-if)#ip address 10.1.1.1 255.255.255.0
GTU_NDN(config-if)#no shut
GTU_NDN(config-if)#end
GTU_NDN#
GTU_NDN#ping 10.1.1.254
Type escape sequence to abort.
Sending 5, 109-byte ICMP Echos to 19.1.1.254, timeout is 2 seconds:
*Sep 28 13:52:40.253: %LINEPROTO-S-UPDOWN: Line protocol on Interface GigabitEth
ernet1, changed state to up.!!
Success rate is 49 percent (2/5), round-trip min/avg/max = 3/5/7 ms
GTU_NDN#sh ip int brief
Interfeoe_1 , LIPeAddreeel , 0K? Method Stotue , Protoool
GigabitEthernet1 10.1.1.1 .VYES manual up              up
GigabitEthernet2 unassigned YES unset administratively down down
GigabitEthernet3 unassigned YES unset administratively down down
GigabitEthernet4 unassigned YES unset administratively down down
GTU_NDN#
```

ნახ. 44. ინტერფეიზე სტატიკური IP მისამართების ნახვა

```
Success rate is 100 percent (5/5), round-trip min/avg/max = 3/3/4 ms
GTU_NDN#conf t
Enter configuration commands, one per line. End with CNTL/Z.
GTU_NDN(config)#int
GTU_NDN(config)#interface virtualpor .
GTU_NDN(config)#interface virtualportGroup ?
    <0-31> VirtualPortGroup interface number

GTU_NDN(config)#interface virtualportGroup 0 1
GTU_NDN(config-if)#
.*Sep 28 13:53:39.127: %IOSXE-4-PLATFORM: R9/9: kernel: dev->name [intsvc0]: dev
_entry not populated
GTU_NDN(config-if)#ip un
GTU_NDN(config-if)#ip un
i*Sep 28 13:53:42.198: %LINEPROTO-S-UPDOWN: Line protocol on Interface VirtualPor
tGroup9, changed state to upn
GTU_NDN(config-if)#ip unnumbered Gig
GTU_NDN(config-if)#ip unnumbered GigabitEthernet 1
GTU_NDN(config-if)#end
GTU_NDN#sh int brief
Interface        IP-Address 0K? Method Status Protocol
GigabitEthernet1 10.1.1.1   YES manual up up
GigabitEthernet2 unassigned YES unset administratively down down
GigabitEthernet3 unassigned YES unset administratively down down
GigabitEthernet4 unassigned YES unset administratively down down
VirtualPortGroup0 10.1.1.1  YES unset up               up
GTU_NDN#
```

ნახ. 45. IP მისამართის ჩამატება ვირტუალურ ჯგუფში

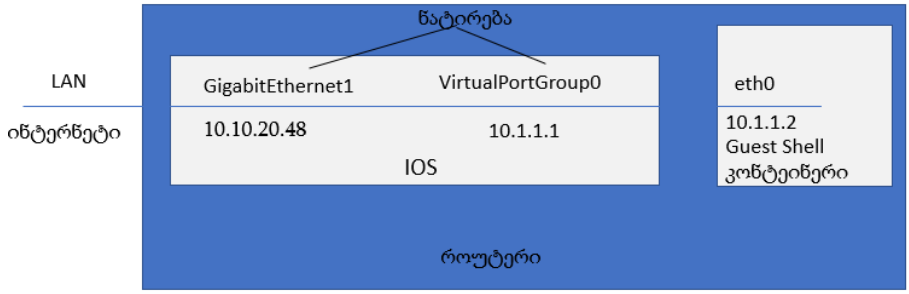
როუტერსა და კონტეინერში გაშვებული ლინუქსის შორის დაკავშირება ხდება შემდეგნაირად:Guestshell enable virtualPortGroup 0 guest-ip 10.1.1.2. ნახ.46 - იხილეთ ვირტუალური ჯგუფის შექმნის მაგალითი.

```
GTU_NDN#guest
GTU_NDN#guestshell ?
    destroy Disable and uninstall the guest shell service package
    disable Disable the guest shell service package
    enable Enable the guest shell service
    run Execute/run program in the guest shell
    <cr>
GTU_NDN#guestshell ena
GTU_NDN#guestshell enable ?
    VirtualPortGroup Virtual port group
    <cr>

GTU_NDN#guestshell enable vir
GTU_NDN#guestshell enable VirtualPortGroup 0 ?
    guest-ip Guestshell IP address
GTU_NDN#guestshell enable VirtualPortGroup 0 gue
GTU_NDN#guestshell enable VirtualPortGroup 0 guest-ip 10.1.1.2
Please wait for completion
```

ნახ. 46. ვირტუალური ჯგუფის შექმნის მაგალითი

როუტერსა და კონტეინერს შორის ip მისამართების დარიგება ხორციელდება ერთსა და იმავე DHCP pool-იდან.ინტერნეტიდან აპლიკაციების გადმოსაწერად და შემდგომ კონტეინერში დასაყენებლად გავუშვათ guestshell enable VirtualPortGroup 0 guest—ip 10.1.1.2 და აუვილებლად მივუთითოთ DNS სერვერი. მოგახსენებთ, რომ სისტემა არ არის გამართული აქვს ბევრი ბაგი და ამიტომ შესაძლოა იმუშაოს ხარვეზებით. ამიტომ მიზანშეწონილია კონტეინერის ლინუქსიდან აპლიკაციების დაყენება. shell run bash-ის ბრძანებით შეგვიძლია დავრწმუნდეთ რომ ნამდვილად ლინუქსი გვაქვს გაშვებული ამისთვის ავკრიფოთ: uname -a და გამოიტანს მიმდინარე ვერსიის სტატუსს. როუტერი გამოყოფს თავისი პულიდან ip მისამართს კონტეინერესთვის. ნახ. 47-ზე მოყვანილია IOS-ის და ლინუქსის კონტეინერები.



ნახ. 47. IOS-ის და ლინუქსის კონტეინერები

კონტეინერიდან ლინუქსის გამოყენება შესაძლებელია DNS ინფორმაციის დამატებით:

echo nameserver 8.8.8.8 | sudo tee —append /etc/resolv.conf.

DNS შემოწმება ხდება შემდეგნაირად:

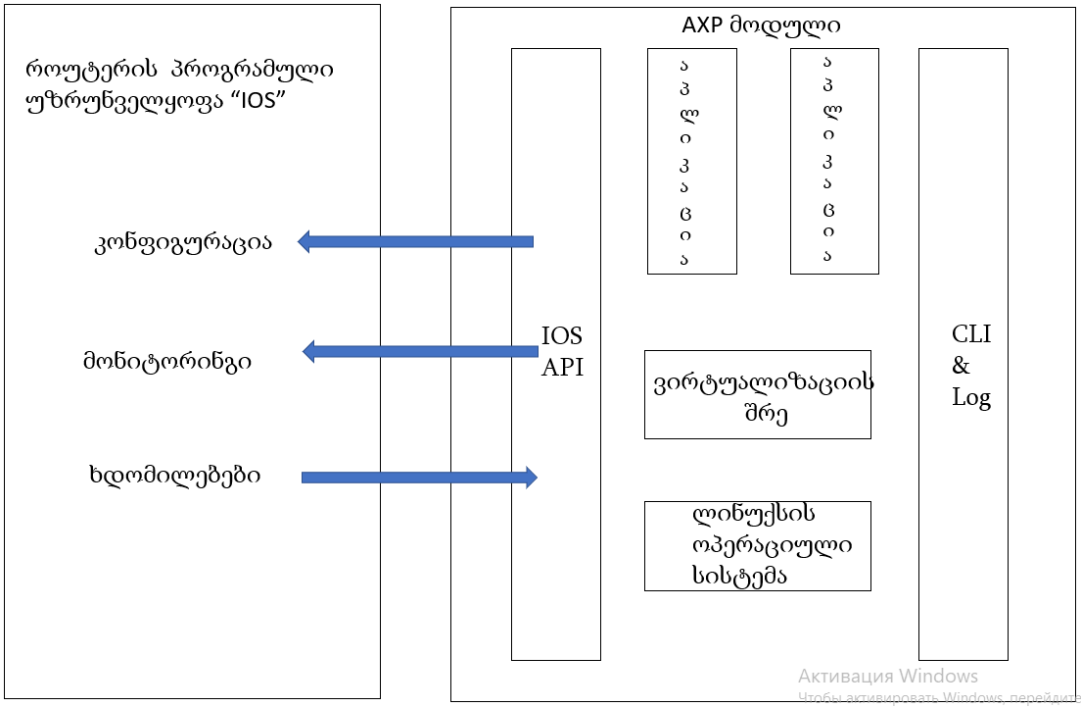
cat /etc/resolv.conf.

შევქმნათ ახალი იუსერი sudo useradd sandro. დავადოთ პაროლი sudo passwd sandro. ვცადოთ ssh პროტოკოლით დავუკავშირდეთ კონტეინერს, ამისთვის ავკრიფოთ ლინუქსის სერვერიდან:ssh sandro@10.1.1.2 და შევიყვანოთ პაროლი. ლინუქსის სერვერიდან გავუშვათ შემდეგი ბრძანება: ssh sandro@10.1.1.2.

კონტეინერიდან აპლიკაციის დაყენება: sudo yum install nano -y

AXP(application eXtension Platform) - შეიძლება ითქვას წარმოადგენს ქსელის ვირტუალიზაციას. შედგება hardware ნაწილისგან, რომელზეც გაშვებულია ლინუქსის ბაზაზე აგებული და მოდიფიცირებული პროგრამული უზრუნველყოფა და პროგმისგან რომელიც განკუთვნილია პროგრამული უზრუნველყოფის შექმნელებისთვის. AXP არის ბარათი რომლის ჩაყენება შესაძლებელია ცისკოს როუტერში. და აქედან გამომდინარე ლინუქსი გაშვებული იქნება ამ ბარათზე და არა ცისკოს CPU, RAM, Flash ან IOS-ში. ზემოთ აღნიშნულ ბარათს ექნება თავისი მეხსიერება, პროცესორი, Flash HD. ბარათი თავსებადია 1841, 2800, 3800 სერიის როუტერებთან. წარმოადგენს სტანდარტულ ქსელის მოდულს NME ან AIM.

AXP-ს აქვს თავისი Linux CLI, ასევე აღსანიშნავია, რომ შეგვიძლია გავუშვათ აპლიკაციები ისეთ ენებზე როგორიც არის: C, Python, Perl, Java. ქვემოთ მოცემულ ნახ.48-ზე ნაჩვენებია AXP სქემა.



ნახ. 48. IOS-ისა და APX მოდულის მოდელი

Cisco IOS და AXP ერთობლივი გამოყენებით მიიღწევა:

-როუტერზე დაკავშირებულ მოწყობილობებთან წვდომა და მათი მართვა

-დინამიური WAN ტუნელის შექმნა

-ქსელის ინტერფეისის აწევა (up/down)

-CLI კომანდების ჩაწერა

-QoS რეგულირება

სხვა სიტყვებით რომ ვთქვათ AXP საშუალებით შეგვიძლია ქსელის მართვა გარკვეულწილად. ქვემოთ მოცემულია Cisco AXP სამი მოდელი, კერძოდ ორი NME ბარათი და ერთი ცალი AIM ბარათი.

AIM ბარათი (AIM-APPRE-102-K9) გთავაზობთ ინტელის 300Mhz პროცესორს, 256MB RAM-ს, და 1GB flash ინფორმაციის საცავს.

NME-APPRE-302-K9 NME ბარათი წარმოადგენს 1Ghz ინტელის Celeron-ის პროცესორს, 512MB RM, and 80GB მყარი დისკით.

NME-APPRE-522-K9 წარმოადგენს 1.4Ghz ინტელის პროცესორს, 2GB RAM და 160GB მყარი დისკით.

აღსანიშნავია, რომ ზემოთ ჩამოთვლილ ბარათებს მოყვება თავისი SKU სოფთი. მაგ.: AIM-APPRE-102-K9, NME-APPRE-302-K9 და NME-APPRE-522-K9.

OpenWRT-ლინუქსის ბაზაზე აგებული სისტემა, რომელიც ინტერნეტში თავისუფალ წვდომაშია. წარმოადგენს ლინუქსის სრულფასოვან დისტრიბუტივს. OpenWRT-ის საშუალებით შეგვიძლია შევასრულოთ ცისკოს IOS XE მსგავსი ქმედებები. ოფიციალურად NFD-ს გააჩნია შემდეგი პლატფორმების მხარდაჭერა:

Ubuntu 18.04 (amd64, armhf, i386)

Ubuntu 20.04 (amd64)

macOS 10.13

macOS 10.14

macOS 10.15

CentOS 8

ასევე NFD დაყენება შესაძლებელია შემდეგ პლატფორმებზე:

Debian 10 (Buster)

Fedora >= 29

Gentoo Linux

Raspbian >= 2019-06-20 (Buster)

აღსანიშნავია, რომ NFD-ის ოფიციალურად არ აქვს ზემოთ აღნიშნული პლატფორმების მხარდაჭერა. თავდაპირველად შევქმანთ PPA საცავი, ამისთვის უნდა გავუშვათ შემდეგი ბრძანება:

sudo apt install software-properties-common

PPA საცავის დაყენების შემდეგ დავამატოთ NDN PPA საცავი:

sudo add-apt-repository ppa:named-data/ppa

sudo apt update

NFD-ის დაყენების ბრძანება:

sudo apt install nfd

არსებობს NFD-ის დაყენების ალტერნატიული ვარიანტი:

git clone https://github.com/named-data/ndn-cxx.git

git clone --recursive https://github.com/named-data/NFD.git

უბუნტუზე NFD-ის გასაშვებად საჭიროა ndn-cxx library-ს დაყენება:

sudo apt install libpcap-dev libsystemd-dev

NFD-ის წყაროდან დასაყენებლად გამოვიყენოთ ქვემოთ ჩამოთვლილი ბრძანებები.

./waf configure # on CentOS, add --without-pch

./waf

sudo ./waf install

NFD-ის გასაშვებად გამოვიყენოთ შემდეგი ბრძანება:

nfd-start

ხოლო გასათიშად კი: nfd-stop

UDP ტუნელის შესაქმნელად NFD-სა და როუტერს შორის ტერმინალში უნდა გავუშვათ შემდეგი ბრძანება:

nfdc face create udp://<other-host>

სადაც <other-host> არის ჰოსტის IP (მაგ.:udp://ndn.example.net). შედეგად ტერმინალში მივიღებთ:

face-created id=308 local=udp4:// 10.0.2.15:6363 remote=udp4:// 131.179.196.46:6363 persistency=persistent.

მარშრუტის დამატებისთვის ბრძანება:

nfdc route add /ndn udp://<other-host>

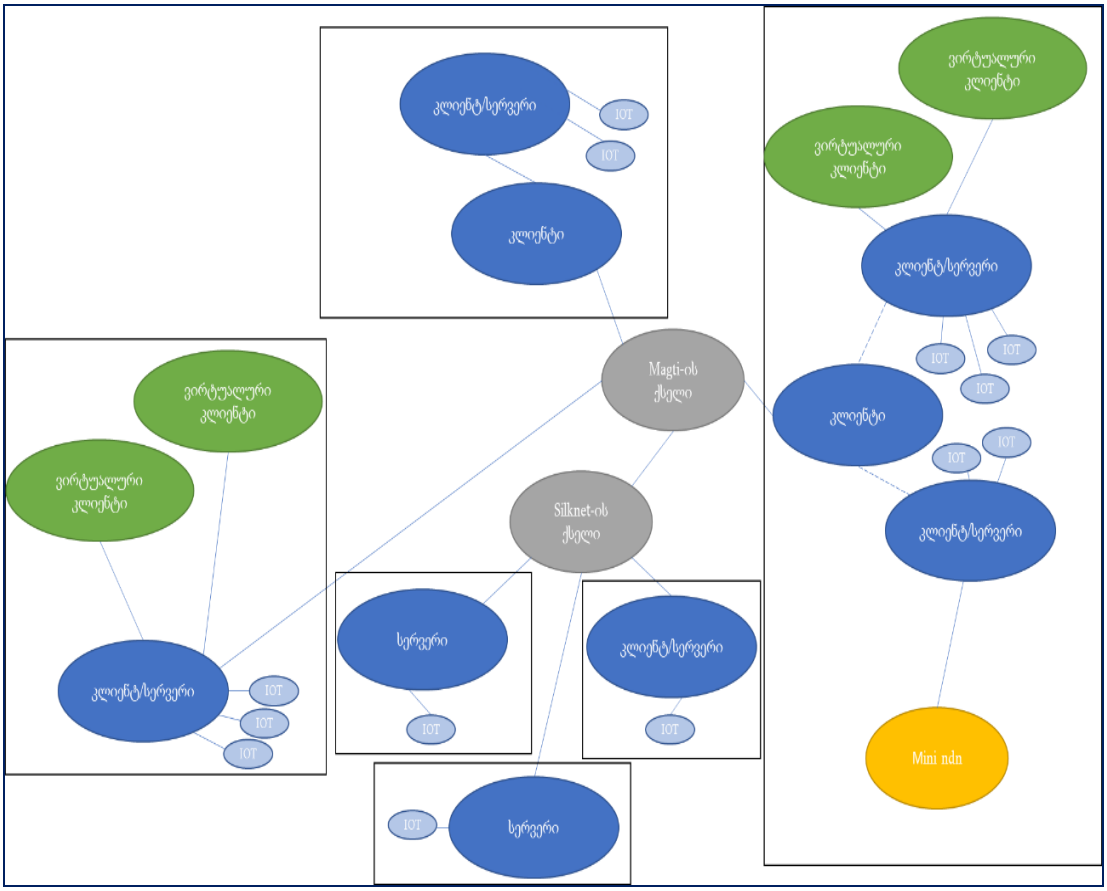
route-add-accepted prefix=/ndn nexthop=308 origin=static cost=0 flags=child-inherit expires=never

NFD-ის საშუალებით ინტერესთა პაკეტი, რომელიც იწყება /ndn-ით იქნება გადამისამართებული შემდეგ ჰოსტამდე face-ის გავლით. მარშრუტები იწერება როგორც სტატიკურად ასევე დინამიურად. აღსანიშნავია, რომ NFD-ს აქვს ფითჩერი, რომელიც უზრუნველყოფს პრეფიქსის გაზიარებას ქსელში.

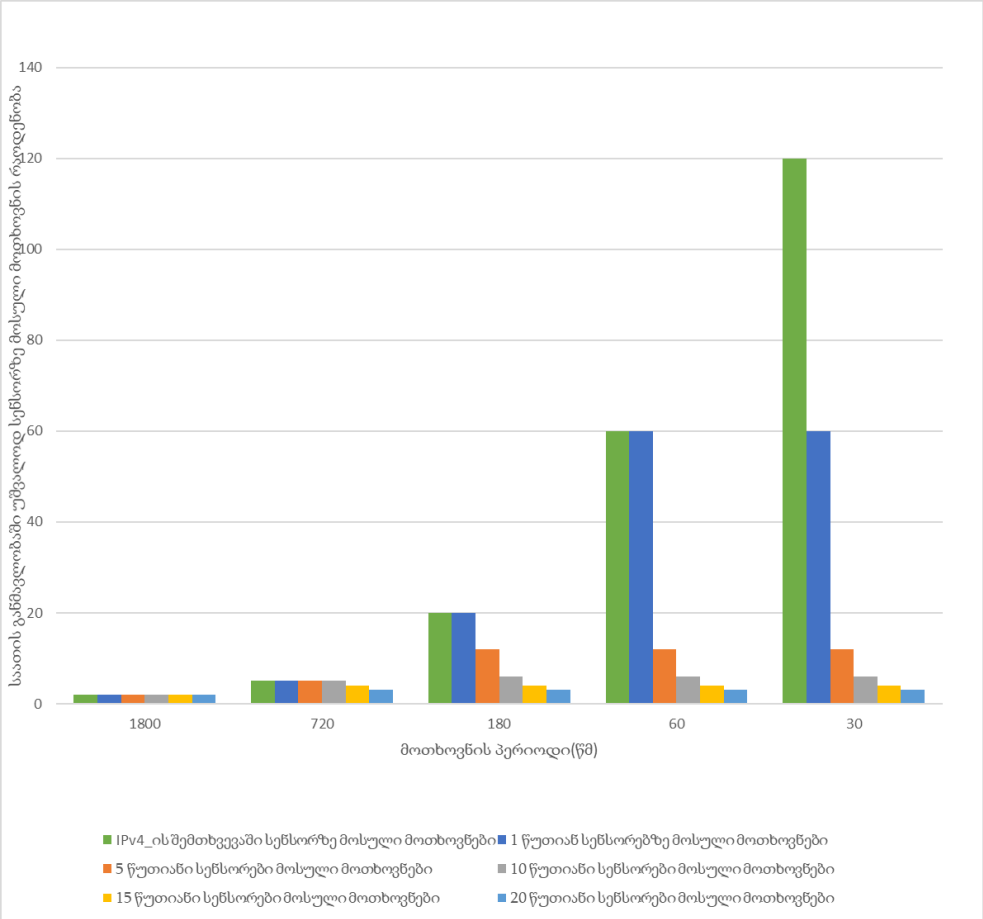
3.5 თბილისის უბანზე მცირე NDN ქსელზე ჩატარებული კვლევა

ჩვენს შემდეგ ექსპერიმენტში თბილისის სხვადასხვა წერტილში გავუშვით NDN ქსელის კვანძები. ასეთი მოწყობილობები ჩართულია სხვადასხვა ოპერატორის IPV4-ის ქსელში და მუშაობენ ენკაპსულაციის პრინციპით. აღებული იყო სხვადასხვა კოპიუტერი რომელზეც დაყენდა მცირედ მოდიფიცირებული NFD-ის ფრეიმვორკი. ჩვენ მიერ შედგემილი ქსელის ტოპოლოგია იხილეთ ნახ.49-ზე. აქ გვაქვს სხვადასხვა კოპიუტერი, რომელიც მუშაობს კლიენტის ან/და სერვერის რეჟიმში. სერვერ მოწყობილობებს აქვთ შესაბამისი ვირტუალური IOT სენსორები. ეს

სენსორები სვადასხვა დროის ინტერვალით აკეთებენ გაზომვას. გათვლა გაკეთებულია ისეთ სენსორებზე, რომლებსაც ენერგიის დაზოგვის მიზნად არ აკეთებენ ხშირ გაზომვებს. სენსორებზე ხშირი მოთხოვნების შემთხვევებში მათი განმეორებითი შეწუხება არ ხდება. ასევე კლიენტების მიერ მოთხოვნილი მონაცემების მიღების დაყოვნება შემცირებულია. Python სკრიპტით პერიოდულად იგზავნება ამ სენსორებზე მოთხოვნა. მოთხოვნა გადის, როგორც IPv4-ის ასევე NDN-ის საშუალებით. ეს მოთხოვნები ხდება პარალერულად. ნახ-50-ზე ნაჩვენებია დატვირთვის პერიოდის და IOT მოწყობილობაზე შემოსული მოთხოვნების კორელაცია.



ნახ. 49. თბილისში აგებული სატესტო NDN ქსელის ტოპოლოგია



ნახ. 50. დატვირთვის პერიოდის და IOT მოწყობილობაზე შემოსული მოთხოვნების კორელაცია

მონაცემების მიხედვით, დატვირთულ პერიოდებში, NDN ქსელი განტვირთავს ამ IOT სენსორებს. იხილეთ ნახ 50. დაყოვნების მაჩვენებლის გაუმჯობესება დამოკიდებულია ორიგინალი სერვერის მდებარეობაზე და რადგან ტესტირება ხდება შედარებით ლოკალურ არეალში მისი გავლენა დიდი არ არის. ტესტით დადგენილია ყველაზე შორეული კვანძებს შორის დაყოვნება 8მწმ. ამ დაყოვნების შემცირების პოტენციალი არის დაახლებით 1მწმ.

დასკვნა

თბილისის მასშტაბით ავაგეთ მცირე NDN ქსელი. როგორც მოგეხსენებათ NDN შემუშავების რეჟიმშია და მისი რეალიზაცია ნამდვილ ქსელში ამ ეტაპზე არ არის. ერთ-ერთ სირთულეს წარმოადგენდა მისი თავსებადობა არსებულ ქსელთან. ჩვენ ეს პრობლემა გადავწყვიტეთ ორნაირი გზით:

1) ჩვენს არსებულ მოწყობილობებზე დავაყენეთ და გავმართეთ NDN-ის ფრეიმვორკი. ასეთ მოწყობილობებს უკვე აქვთ შესაძლებლობა ქსელურ დონეზე დაამუშავონ როგორც IPv4-ის პაკეტი, ასევე NDN-ის პაკეტი.

2) ქსელში არსებულ სხვა მოწყობილობებზე თავსებადობის პრობლემა გადავჭერით NDN პაკეტის IPv4-ის ენკაფსულაციით. ამ მოწყობილობებისთვის არაფერი არ იცვლება, მაგრამ რაც მთავარია ხელს არ უშლიან NDN-ის არსებობაში.

სატესტო ქსელში, ავტომატიზირებული საშუალებით, პერიოდულად იგზავნებოდა NDN-ის და IPv4-ის მონაცემები. კლიენტები ითხოვდნენ ერთი და იგივე მონაცემებს როგორც IPv4-ის საშუალებით, ასევე NDN-ის საშუალებით. მონაცემის წყაროდ გვაქვს აღებული ვირტუალური IoT სენსორები, რომლებსაც აქვთ მონაცემთა აქტუალობის სხვადასხვა პერიოდი. შედეგების მიხედვით დატვირთულ პერიოდებში NDN ქსელი რამოდენიმეჯერ განტვირთავს ამ სენსორებს. მოთხოვნა მიდის უახლოეს NDN მოწყობილობაზე და IoT მოწყობილობის შეწუხება აღარ ხდება. ასეთი IoT მოწყობილობები შეიძლება იყოს ქუჩებში დაყენებული ტემპერატურის, გამონაბოლქვის, რადიაციის, ვირუსების სენსორები, რომლებიც გაზომვას აკეთებენ 20 წუთში ერთხელ(ენერგიის დასაზოგად), მაგრამ მონაცემზე მოთხოვნა ხდება ხშირად. კიდე დრონები რომლებიც პერიოდულად იღებენ აერო-ფოტოს და საზოგადოებრივ ადგილებში ხალხის მოცულობის მთვლელი მოწყობილობები; სენსორების და ქსელის კვანძების განტვირთვა ხდება მარტო დიდი დატვირთვის შემთხვევებში. ეს ხდება ზუსტად მაშინ

როდესაც ამაში არის საჭიროება. გარდა ამისა, არსებულ IPv4-ის ქსელებთან შედარებით, ჩვენ მიერ წარმოდგენილ ქსელში მონაცემების მიღების დაყოვნება შემცირებულია. განსაკუთრებით მაშინ, როდესაც მონაცემების გადაცემა ხდება დიდი ინტენსივობით.

ნაშრომში ჩატარებული კვლევები და მიღებული შედეგები შეიძლება ასე ჩამოვაყალიბოთ:

1) განხილულია და გაანალიზებულია NDN არქიტექტურა, მისი უსაფრთხოების და თავსებადობის საკითხები არსებულ ქსელებთან.
2) ჩატარებულია ექსპერიმენტი, სადაც NDN სატესტო ქსელის საშუალებით მივაღწიეთ დაყოვნების შემცირება საშუალოდ 60%-ით. აღსანიშნავია, რომ თავდაპირველად NDN-ის შემთხვევაში დაყოვნება გაიზარდა დაახლოებით 17%-ით, ვინაიდან პირველი პაკეტის დაქეშირების დროს სიგნალის დამუშავებას დაჭირდა გარკვეული პერიოდი.
3) გამოკვლეულია NLSR-ის დაყოვნება პაკეტების ხელმოწერითა და ხელმოწერის გარეშე. გამოყენებული ნდობის მოდელის პირობებში, რომელიც მოითხოვს გასაღების მრავალდონიან ვერიფიკაციას, გასაღების აუტენტიფიკაცია დამატებით ზრდის დატვირთვას საშუალოდ 20-25%-ით მხოლოდ LSA-ის 4 განსაზღვრულ ეტაპში.
4) ჩატარებულია ექსპერიმენტი, სადაც მიღწეულია სენსორების ენერგომოხმარების შემცირება. ერთ-ერთ ჩატარებულ ექსპერიმენტში ენერგომოხმარება დაეცა 23,7%-ით. ენერგომოხმარების მნიშვნელობა გამოხატულია ფორმულით.

გამოყენებული ლიტერატურა

1. GLOBAL INFORMATION INFRASTRUCTURE, INTERNET PROTOCOL ASPECTS, NEXT-GENERATION NETWORKS, INTERNET OF THINGS AND SMART CITIES; Series Y Supplement 47; 04.2018, pp 5-6.

2. GLOBAL INFORMATION INFRASTRUCTURE, INTERNET PROTOCOL ASPECTS, NEXT-GENERATION NETWORKS, INTERNET OF THINGS AND SMART CITIES; Series Y Supplement 47, 04.2018, p 23.

3. Beichuan Zhang, Lan Wang - A Brief Introduction to Named Data Networking, 01.2019, 7p.

4. https://en.wikipedia.org/wiki/Named_data_networking, უკანასკნელად იქნა გადამოწმებული - 05.28.2021

5. Lixia Zhang, Allison Mankin - NDNS : A DNS-Like Name Service for NDN, 2017, 9p.

6. https://named-data.net/publications/tutorials/, უკანასკნელად იქნა გადამოწმებული - 10.05.2019.

7. http://www.sharetechnote.com/html/IoT/IoT_Definition.html, უკანასკნელად იქნა გადამოწმებული - 09.08.2020.

8. http://www.sharetechnote.com/html/IoT/IoT_WhatToChoose.html, უკანასკნელად იქნა გადამოწმებული - 09.08.2020.

9. https://www.sharetechnote.com/html/IoT/LR_LPWAN_LoRa.html, უკანასკნელად იქნა გადამოწმებული - 09.08.2020.

10. Mahendra Sapkota, Takuro Sato Vehucular Communication Using Named Data Networking, 04.12.2017, 4p.

11. Greg White, Greg Rutz, CONTENT DELIVERY WITH CONTENTCENTRIC NETWORKING, February 2016, pp 7-8.

12. P. Dell’aversana. Artificial Neural Networks and Deep Learning, 15.05.2020, 14p.

13. Alexander Afanasyev, Junxiao Shi, Beichuan Zhang, Lixia Zhang, Ilya Moiseenko, Yingdi Yu, Wentao Shang, Yanbiao Li, Spyridon Mastorakis, Yi Huang, Jerald Paul Abraham, Eric Newberry, Steve DiBenedetto, Chengyu Fan, Christos Papadopoulos, Davide Pesavento, Giulio Grassi, Giovanni Pau, Hang Zhang, Tian Song, Haowei Yuan, Hila Ben Abraham, Patrick Crowley, Syed, Obaid Amin, Vince Lehman, Muktadir Chowdhury, and Lan Wang, NFD Developer's Guide Revision 7 (October 4, 2016). 72p.

14. Vince Lehman, Muktadir Chowdhury, Nicholas Gordon, Ashlesh Gawande, University of Memphis, NLSR Developer's Guide (November, 2017). 74.p

15. Amar Abane, Mehammed Daoui, Samia Bouzefrane, Soumya Banerjee, Paul Muhlethaler, A realistic named data networking architecture for the Internet of things , 11.2019. 28p.

16. https://it.wikipedia.org/wiki/IPv4, უკანასკნელად იქნა გადამოწმებული - 05.29.2021.

17. Todd Lammle, CCNA Routing and Switching Study Guide, 2019. pp 102-105.

18. ზანგალაძე ა.პ., სახელდარქმეული მონაცემთა ინტერნეტი და მისი უპირატესობა. საინჟინრო სიახლენი, 2021, №1, vol. 92,გვ. 57-62;

19. Zangaladze A. P., Kvernadze S. A., Kvirkvelia S. V. INFORMATION-CENTRIC NETWORKING AND ITS OFFLOAD BENEFITS FOR IOT REMOTE DEVICES. Научные Горизонты, 2021, №5(45), pp. 132-138;

20. კვერნაძე მ.ა., ზანგალაძე ა.პ., კვირკველია შ.ვ., კვერნაძე ს.ა., ბერიძე ჯ.ლ. რადიოსიგნალების დამუშავების ადაპტიური მოდელი შემდეგი თაობის რადიოსისტემებისთვის. საქართველოს საინჟინრო სიახლენი, 2021, №1, vol. 92, გვ. 45-56;