საქართველოს ტექნიკური უნივერსიტეტი

ხელნაწერის უფლებით

ალექსანდრე ზანგალაძე

უსადენო კავშირის ტექნოლოგიების საგნების ინტერნეტისთვის გამოყენების პრობლემების კვლევა

დოქტორის აკადემიური ხარისხის მოსაპოვებლად

წარდგენილი დისერტაციის

ა ვ ტ ო რ ე ფ ე რ ა ტ ი

სადოქტორო პროგრამა: ციფრული სატელეკომუნიკაციო ტექნოლოგიები

შიფრი: 0714

თბილისი

2021 წელი

სამუშაო შესრულებულია საქართველოს ტექნიკურ უნივერსიტეტში
ენერგეტიკისა და ტელეკომუნიკაციის ფაკულტეტი
ტელეკომუნიკაციის დეპარტამენტი

ხელმძღვანელი: ასოც. პროფესორი შ. კვირკველია

რეცენზენტები:

დაცვა შედგება 2021 წლის ”--------” ----------, -------- საათზე
საქართველოს ტექნიკური უნივერსიტეტის ენერგეტიკისა და
ტელეკომუნიკაციის ფაკულტეტის სადისერტაციო საბჭოს კოლეგიის
სხდომაზე, კორპუსი VIII, აუდიტორია
მისამართი: 0166, თბილისი, კოსტავას 77.

დისერტაციის გაცნობა შეიძლება სტუ-ის ბიბლიოთეკაში,
ხოლო ავტორეფერატისა - ფაკულტეტის ვებგვერდზე

სადისერტაციო საბჭოს მდივანი, გ. გიგინეიშვილი
ასოცირებელი პროფესორი

სამუშაოს ზოგადი დახასიათება

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.

დისერტაციის სტრუქტურა და მოცულობა: დისერტაცია შედგება შესავალისგან, სამი თავის, დასკვნისა და ლიტერატურის ნუსხისაგან. დისერტაციის საერთო მოცულობა შეადგენს 115 გვერდს, 2 ცხრილისა და 50 ნახაზის ჩათვლით.

სამუშაოს ძირითადი შინაარსი

პირველ თავში განხილულია:

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

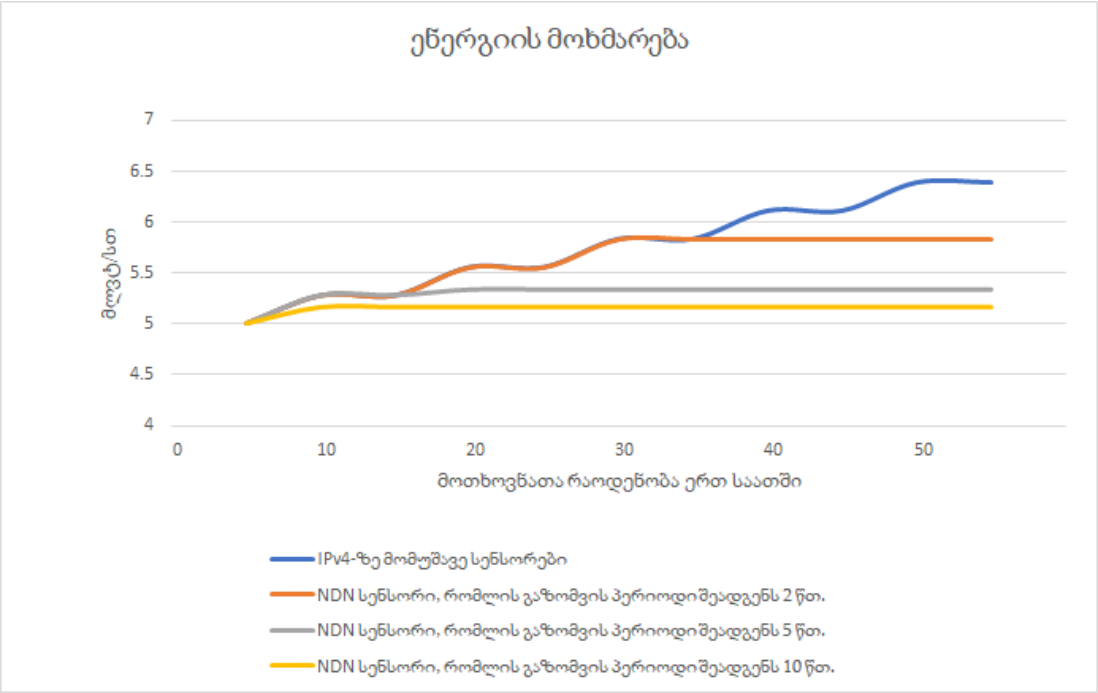
Routing Protocol) მარშრუტიზაციის პროტოკოლი. ამ პროტოკოლით ხდება მარშრუტიზატორებს შორის კონტენტის მისამართების გაზიარება[14].

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:

სატესტო 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 საშუალოად 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.1-ზე წარმოდგენილია ენერგიის მოხმარების გრაფიკი.



ნახ.1 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 სენსორდა 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 ენერგომოხმარება დაიზოგა, მიუხედავად გაზრდილი მოთხოვნათა რიცხვისა.

პირველი 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.

მეორე თავში განხილულია:

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.

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

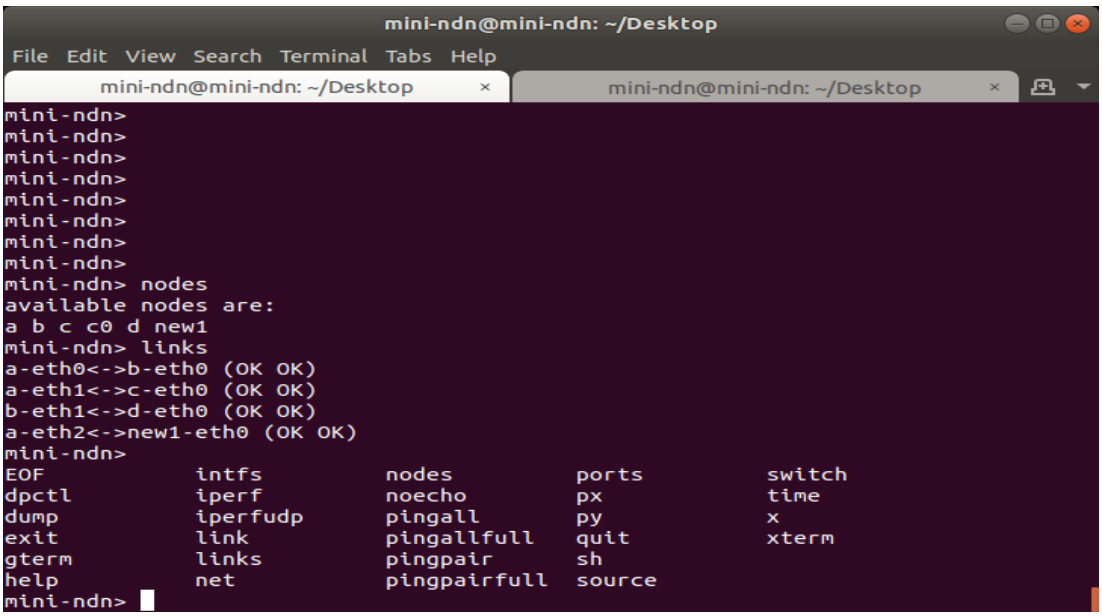
c ndnping -c 4 /ndn/a-site/a/pingtest NDN-პინგის გაგზავნა შესაბამისი კონტენტის მითითებით

d ip route show აი პი 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.2-ზე იხილეთ Mini-NDN-ის სამუშაო გარემო.



ნახ.2 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

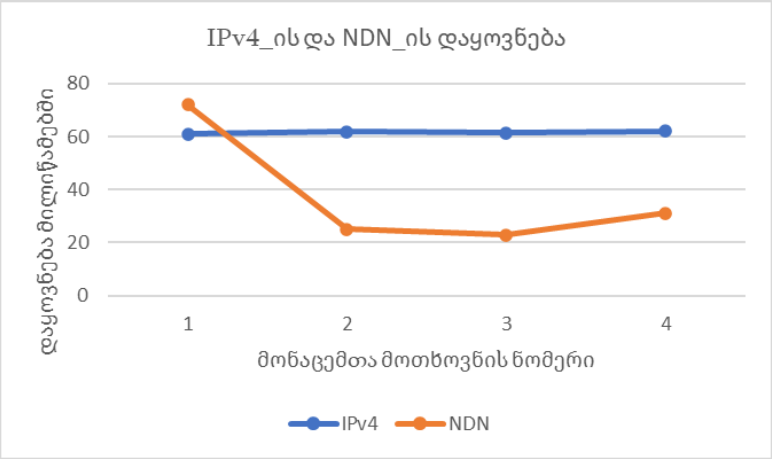
ენკაპსულაციის გათვალისწინებით. ეს იმას ნიშნავს რომ NDN-ის ჰოსტიდან UDP/IP მომუშავე როუტერზე მოსული პაკეტი გაიგზავნება შესაბამისი ჰოსტის მიმართულებით, ანუ ის როუტერი რომლისთვისაც არ იყო განსაზღვრული მონაცემთა პაკეტი, ვერ წაიკითხავს ინტერესთა/მონაცემთა პაკეტის სტრუქტურას, მას უბრალოდ ეცოდინება საით უნდა გააგზავნოს მონაცემები. არსებული ქსელი გარკვეულწილად შეასრულებს მონაცემთა პაკეტის ტრანსპორტირების ფუნქციას.

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 გრაფიკებზე: ნახ. 3, ნახ. 4 და ნახ. 5.

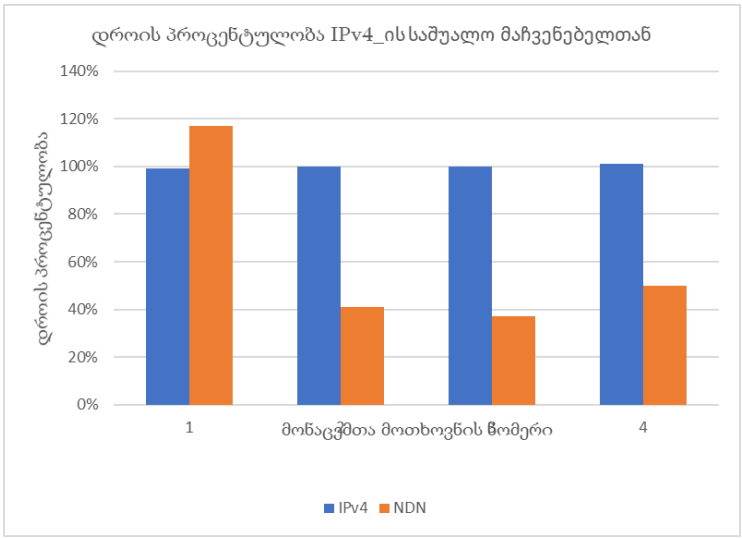
ქვემოთ მოყვანილია Mini-NDN-ის ტერმინალიდან გაშვებული პინგი TCP/IP და NDN შემთხვევებისთვის.

```
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
```

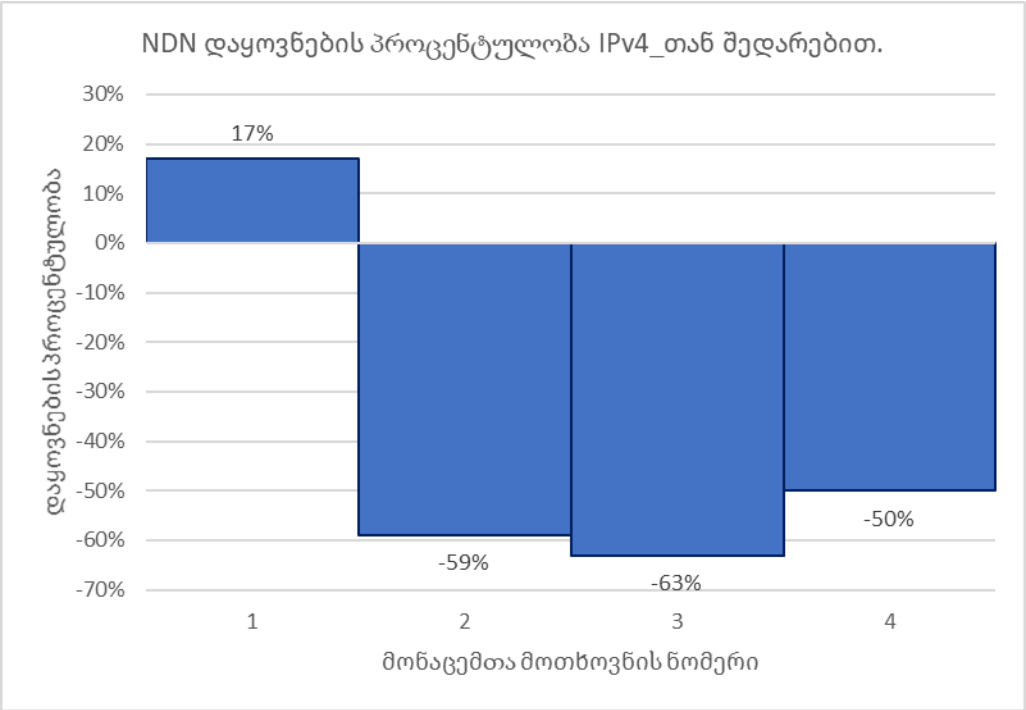
```
mini-ndn> c ndnping_cache -c 4 /ndn/d-site/d/pingtest
PING /ndn/d-site/d/pingtest
content from /ndn/d-site/d/pingtest: seq=15949988448206744268 time=71.9338 ms
content from /ndn/d-site/d/pingtest: seq=15949988448206744269 time=24.9698 ms
content from /ndn/d-site/d/pingtest: seq=15949988448206744270 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
```



ნახ.3. IPv4/NDN დაყოვნება



ნახ.4. დროის პროცენტულობა IPv4 საშუალო მაჩვენებელთან

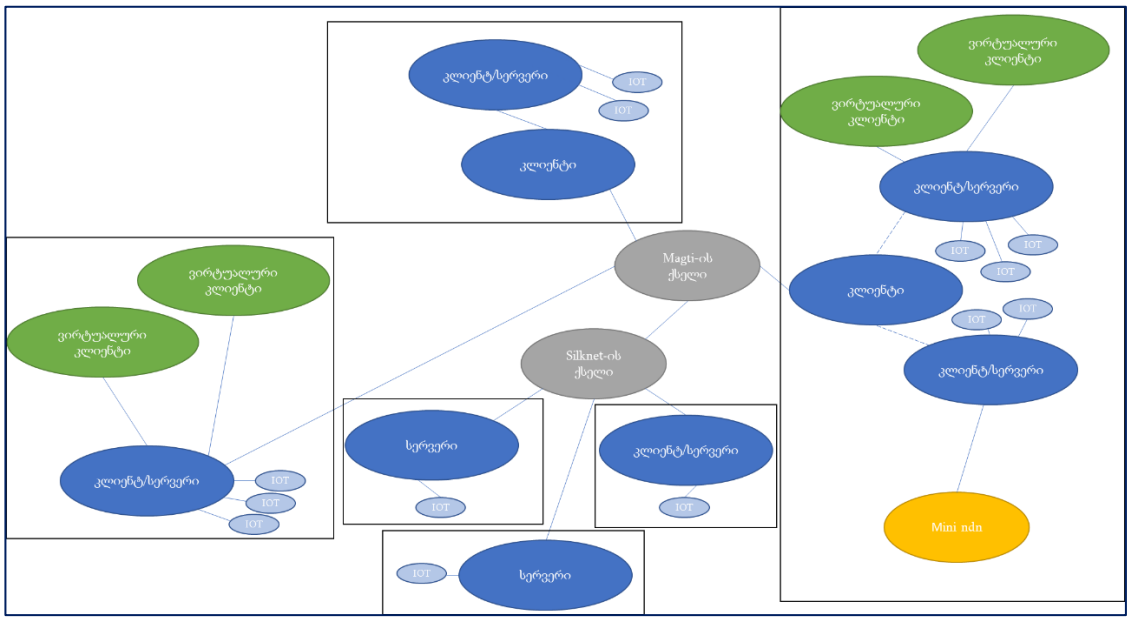


ნახ.5. NDN დაყოვნების პროცენტულობა IPv4 შედარებით

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

მესამე თავში განხილულია:

ჩვენ შემდეგ ექსპერიმენტში თბილისის სხვადასხვა წერტილში გავუშვით NDN ქსელის კვანძები. ასეთი მოწყობილობები ჩართულია სხვადასხვა ოპერატორის IPV4-ის ქსელში და მუშაობენ ენკაპსულაციის პრინციპით. აღებული იყო სხვადასხვა კომპიუტერი რომელზეც დაყენდა მცირედ მოდიფიცირებული NFD-ის ფრეიმვორკი. ჩვენი სატესტო ქსელის ტოპოლოგია წარმოდგენილი ნახ. 6-ზე.

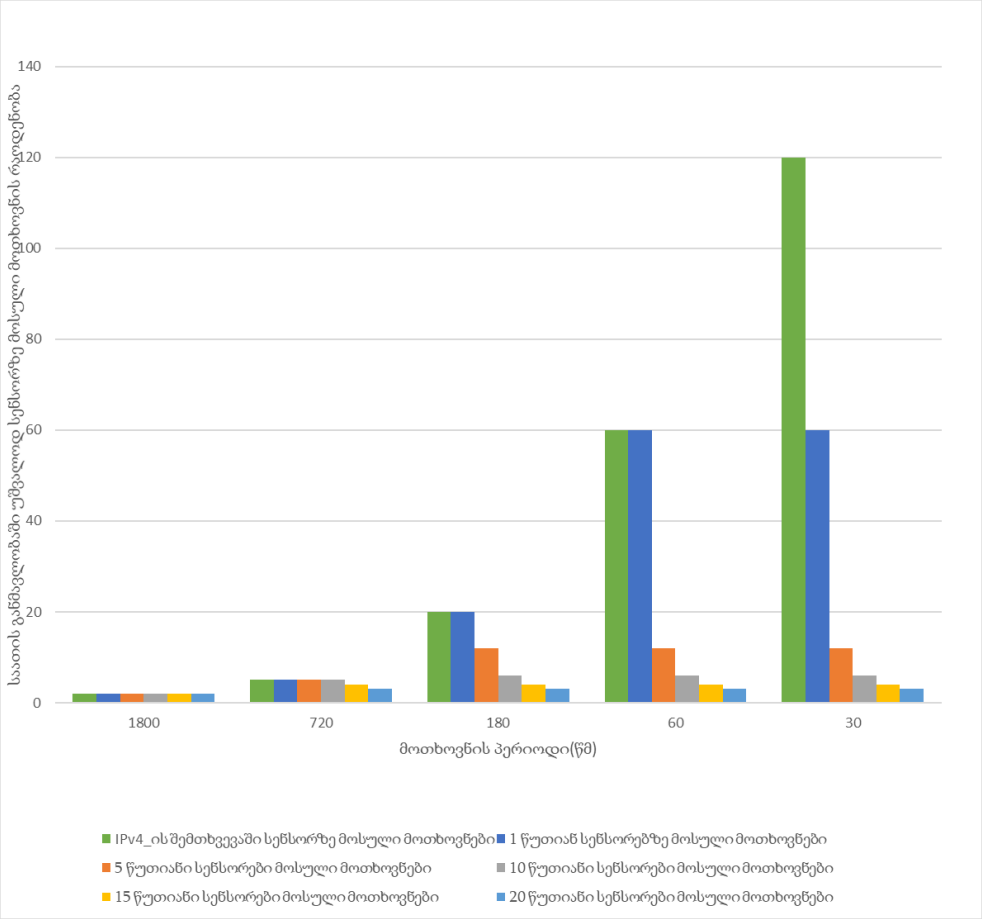


ნახ.6. წარმოდგენილია თბილისში აწყობილი სატესტო NDN ქსელის ტოპოლოგია

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-ნ სკრიპტით პერიოდულად იგზავნება ამ სენსორებზე მოთხოვნა. მოთხოვნა გადის, როგორც IPv4-ის ასევე NDN-ის საშუალებით. ეს მოთხოვნები ხდება პარალერულად. ნახ.2-ზე ნაჩვენებია დატვირთვის პერიოდის და IOT მოწყობილობაზე შემოსული მოთხოვნების კორელაცია.

მონაცემების მიხედვით, დატვირთულ პერიოდებში, NDN ქსელი ჯერებზე განტვირთავს ამ IOT სენსორებს. 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.



ნახ.7. დატვირთვის პერიოდის და IOT მოწყობილობაზე შემოსული მოთხოვნების კორელაცია

დასკვნა

თბილისის მასშტაბით ავაგეთ მცირე NDN ქსელი. როგორც მოგეხსენებათ NDN შემუშავების რეჯიმშია 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.

გამოქვეყნებული ნაშრომების სია დისერტაციის თემაზე:

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

2. 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;

3. 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;

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.