ო. ქართველიშვილი, მ. ქართველიშვილი

მარშრუტიზაციის ალგორითმები კომპიუტერულ ქსელებში

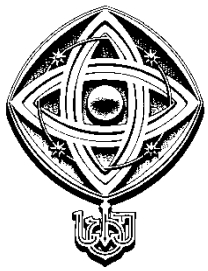
#ტექნიკური უნივერსიტეტი@

საქართველოს ტექნიკური უნივერსიტეტი

ო. ქართველიშვილი, მ. ქართველიშვილი

მარშრუტიზაციის ალგორითმები კომპიუტერულ ქსელებში

დამტკიცებულია სტუ-ს სარედაქციო-საგამომცემლო საბჭოს მიერ



თბილისი
2006

შაკ 681&3

გამოკვლეულია მარშრუტიზაციის მოქმედება ზოგად ქსელებში, კერძოდ ქსელების ამჟამად ყველაზე გავრცელებულ ოჯახში - **ესზჰშზ** ქსელებში, როგორიც არის ინტერნეტი. ნაჩვენებია მათი დადებითი და უარყოფითი მხარეები, განხილულია მათ მუშაობასთან დაკავშირებული პრობლემები და მათი გადაჭრის გზები. განხილულია რამოდენიმე ყველაზე მნიშვნელოვანი ალგორითმი და ამ ალგორითმის ბაზაზე 4 ყველაზე ხმარებადი პროტოკოლი: ორი შიდა მარშრუტიზაციის - **კშზ** და **ფთზა**, ორი გარეშე მარშრუტიზაციის პროტოკოლი - **შპზ** და **იპზ**. განსაზღვრულია მათი გამოყენების სფეროები. მოყვანილია ალგორითმების საილუსტრაციო მაგალითები.

დამხმარე სახელმძღვანელო გათვალისწინებულია “კომპიუტერული სისტემების და ქსელების„ სპეციალობის (22-10) სტუდენტებისა და მაგისტრანტებისათვის.

რეცენზენტები: პროფ. კ. კამკამი□ე,

პროფ. ნ. ლომინა□ე

ISBN 99940-57-15-4

შესავალი

მარშრუტიზაციის ალგორითმის ქვეშ ხშირად იგულისხმება ქსელის დონის პროტოკოლი, რომელიც მართავს პაკეტებს მათი გადაადგილების დროს ქსელში დანიშნულების ადგილამდე. დროის მომენტები, როდესაც მიიღება გადაწყვეტილებები მარშრუტის არჩევის შესახებ, დამოკიდებულია იმაზე, ქსელის მიერ გამოიყენება დეიტაგრამული გადაცემა, თუ ვირტუალური შეერთებებით მუშაობა. დეიტეგრამულ ქსელებში მარშრუტიზაციის პროტოკოლის დახმარებით მიმდევრობითი პაკეტები ერთი და იგივე წყვილისათვის შეიძლება გადაიცენ სხვადასხვა მარშრუტით და მარშრუტის არჩევა თვითეული პაკეტისათვის უნდა მოხდეს ინდივიდუალურად. ქსელებისათვის ვირტუალური შეერთებით მარშრუტის არჩევა ხდება თვითოეული ვირტუალური შეერთების შემთხვევაში. მარშრუტიზაციის ალგორითმი გამოიყენება ქსელში გზის არჩევისათვის მოცემულ ვირტუალურ შეერთებისათვის. ვირტუალური შეერთების პაკეტები მიმდევრობით გამოიყენებენ ამ გზას იქამდე, სანამ ეს ვირტუალური შეერთება შეწყვეტს თავის არსებობას, ან როცა მოცემული შეერთებისათვის რაიმე მიზეზების გამო აირჩევა სხვა მარშრუტი.

ჩვეულებრივად, მარშრუტის არჩევისათვის გამოიყენება საკმაოდ რთული ალგორითმების სიმრავლე, რომელებიც მუშაობენ მეტ-ნაკლებად დამოუკიდებლად, თუმცა ქსელის კვანძებს შორის ინფორმაციის გაცვლით. ამ ალგორითმების სირთულე განპირობებულია მთელი რიგი მიზეზით. უპირველეს ყოვლისა, მარშრუტიზაცია მოითხოვს ქსელის ყველა კვანძის მუშაობის კოორდინაციას, მეორეც, მარშრუტიზაციის სისტემა უნდა რეაგირებდეს არხების ან კვანძის მწყობრიდან გამოსვლაზე მონაცემთა ნაკადის სხვა მიმართულებით გადაცემით და სამარშრუტო ინფორმაციის განახლებით. მესამე, საუკეთესო მახასიათებლების მისაღწევად მარშრუტიზაციის ალგორითმმა შეიძლება შეცვალოს მარშრუტი, როდესაც ქსელის ზოგიერთი მონაკვეთის მახასიათებლები იცვლებიან.

მოცემულ ნაშრომში ძირითადი ყურადღება ეთმობა მარშრუტიზაციის ამოცანის ორ ასპექტს. პირველი ეხება საუკეთესო მახასიათებლების მქონე მარშრუტის არჩევას. განიხილება უმოკლესი გზის ალგორითმები, რომლებიც ფართოდ გამოიყენებიან პრაქტიკაში. აღწერილია ალგორითმები, რომლებიც შესაძლებლობას იძლევიან მიღწეული იქნას ოპტიმალურთან მიახლოვებული შედეგები. მარშრუტიზაციის ამოცანის მეორე ასპექტი, რომელსაც ეთმობა ყურადღება, მდგომარეობს ქსელის ყველა კვანძებს შორის მარშრუტის არჩევისათვის საჭირო სამარშრუტო ინფორმაციის გავრცელებას.

ორ ძირითად ფუნქციას, რომელთაც ასრულებენ მარშრუტიზაციის ალგორითმები წარმოადგენენ მარშრუტების არჩევა სხვადასხვა წყვილისათვის გამგზავნი-ადრესატი და შეტყობინობების სწორი მიწოდების უზრუნველყოფა ადრესატამდე იმის შემდეგ, რაც მარშრუტი არჩეულია. მეორე ფუნქცია უზრუნველყოფილია სხვადასხვა პროტოკოლებისა და მონაცემთა სტრუქტურების გამოყენებით. ნაშრომში ძირითადი ყურადღება ეთმობა პირველ ფუნქციას და ქსელის ფუნქციონირებაზე მის ზემოქმედებას.

არსებობს მარშრუტიზაციის ალგორითმების კლასიფიკაციის რამოდენიმე ხერხი. ერთ-ერთი მათგანი გულისხმობს ყველა ალგორითმების დაყოფას ცენტრალიზებულად და განაწილებულად. ცენტრალიზებულ ალგორითმებში ყველა მარშრუტის არჩევა ხორციელდება მხოლოდ ცენტრალურ კვანძში, ხოლო

განაწილებულ ალგორითმებში – ქსელის ყველა კვანძებში. ამასთან კვანძები საჭიროების შემთხვევაში ცვლიან ერთმანეთს შორის ინფორმაციას. მარშრუტიზაციის ალგორითმების სხვა კლასიფიკაცია ემყარება იმას, იცვლება თუ არა მარშრუტი ქსელის რიცხობრივ მახასიათებლებთან ერთად. მარშრუტიზაციის სტატიკურ ალგორითმებში გამგზავნი–ადრესატის თვითეული წყვილის გზა ფიქსირებულია და არ არის დამოკიდებული ტრაფიკის მერყეობაზე. იგი შეიძლება შეიცვალოს მხოლოდ რომელიმე კვანძის ან ხაზის მწყობრიდან გამოსვლის შემდეგ. მარშრუტიზაციის ასეთი წესი რეკომენდირებულია გამოყენებული იქნას ან მეტად მარტივი ქსელებისათვის, ან როდესაც ქსელის მუშაობის ეფექტურობა არ არის არსებითი. პაკეტების კომუტაციის მქონე ქსელების დიდი უმრავლესობისათვის გამოიყენება ადაპტიური მარშრუტიზაციის სხვადასხვაობა, რომლის დროსაც გზები გამგზავნიდან ადრესატამდე ახალი ტრაფიკისათვის იცვლება ქსელის დატვირთვის მიხედვით. ამ შემთხვევაში მარშრუტიზაციის ალგორითმი უნდა შეეცადოს შეცვალოს მარშრუტი და მიმართოს ნაკადი გადატვირთული სეგმენტების გვერდის ავლით.

ამჟამად გამოიყენებიან დიდი რაოდენობის მარშრუტიზაციის ალგორითმები, რომლებიც განსხვავდებიან ერთმანეთისაგან სირთულით და ეფექტიურობით.

1. მარშრუტიზაციის ალგორითმების კლასიფიკაცია და მათი პარამეტრები

მარშრუტიზაციის ალგორითმის ტიპი და რეალიზაციის პრინციპები ძირითადად იცვლება გადასაწყვეტი ამოცანის მიხედვით. არსებობს მარშრუტიზაციის ალგორითმების შეფასების შემდეგი პარამეტრები:

ალგორითმის ოპტიმალურობა. ეს პარამეტრი ახასიათებს ალგორითმის უნარს ამოირჩიოს საუკეთესო მარშრუტი. ამ უკანასკნელის არჩევის მექანიზმი დამოკიდებულია კრიტერიუმთა სიმრავლეზე და თითოეული ამ კრიტერიუმის წონაზე.

მცირე გამოთვლითი დანახარჯები. ალგორითმი უნდა იყოს რაც შეიძლება მარტივი, რათა მაქსიმალურად უზრუნველყოფდეს ფუნქციონალურ შესაძლებლობებს და ამავე დროს ზოგავდეს გამოთვლით რესურსებს, არ აყენებდეს მაღალ მოთხოვნებს აპარატული და პროგრამული უზრუნველყოფის წინაშე.

მუშაობის სტაბილურობა. ალგორითმი უნდა შეუფერხებლად მუშაობდეს არახელსაყრელი და მოულოდნელი სიტუაციების პირობებშიც, როგორიცაა ქსელის მაღალი დატვირთვა, მონაცემთა არაკორექტული გადაცემა, ქსელის ტოპოლოგიის სწრაფი ცვლა და ა.შ. ალგორითმის სტაბილურობას ძალიან დიდი მნიშვნელობა აქვს, რადგან მარშრუტიზატორები განლაგებულნი არიან ქსელის კვანძებში და მათი მწყობრიდან გამოსვლა იწვევს მთელი ქსელის მუშაობაში საგრძნობ პრობლემებს. ძირითადად სტაბილურად მიიჩნევა ის ალგორითმი, რომელიც დროით გამოიცდება.

სწრაფი კრებადობა. **კრებადობა** - ეს არის მარშრუტიზატორების ოპტიმალურ მარშრუტებზე შეთანხმების პროცესი. როცა ქსელში რაღაც მოვლენას აქვს ადგილი, მაშინ ან ჩნდება ახალი ან ისპობა ძველი მარშრუტები. მარშრუტიზატორები ერთმანეთს ატყობინებენ ამ ცვლილებების შესახებ, რაც იწვევს ოპტიმალური მარშრუტების გადათვლას და საბოლოოდ მარშრუტიზატორები უნდა მივიდნენ ერთიან შეთანხმებაზე. თუ ეს პროცესი ნელა ხორციელდება, შესაძლებელია სამარშრუტო მარყუჟების წარმოშობა, რაც გამოიწვევს ქსელის მუშაობის შეფერხებას.

მოქნილობა. რადგან ქსელი განუწყვეტლივ იცვლება, ალგორითმს უნდა ჰქონდეს უნარი ადაპტირება მოახდინოს ამ ცვილიებების მიმართ, ე.ი. გაითვალისწინოს რაიმე სეგმენტის მწყობრიდან გამოსვლა ან მისი გამტარუნარიანობის, დაყოვნების, რიგის სიგრძის ცვლილება და სწრაფად გადააგზავნოს ამ სეგმენტით მიმავალი მონაცემთა ნაკადი ამ ცვლილებების გათვალისწინებით საუკეთესო გზით.

მარშრუტიზაციის ალგორითმები შესაძლებელია შემდეგნაირად იქნას კლასიფიცირებული:

სტატიკური და დინამიური. სტატიკური მარშრუტიზაციის ალგორითმები წარმოადგენენ მარშრუტების სტატიკურ ტაბულებთან მუშაობის წესებს. სტატიკური მარშრუტების ტაბულები დგებიან ქსელის ადმინისტრატორის მიერ მარშრუტიზაციის პროცესის დაწყებამდე. სტატიკური მარშრუტიზაციის ალგორითმები არ არის რთული და გამოსადეგია ისეთ ქსელებში, სადაც ქსელის ტოპოლოგია და მონაცემთა ნაკადის სტრუქტურა მარტივია. ამ ტიპის ალგორითმებს არ შეუძლიათ ადაპტირება ქსელში მომხდარი ცვლილებების მიმართ.

დინამიური მარშრუტიზაციის ალგორითმები რეალურ დროში რეაგირებენ ქსელში მომხდარ ცვლილებებზე. შეტყობინებები ამ ცვლილებების შესახებ განჭოლავენ ქსელს და იწვევენ სამარშრუტო ტაბულების გადათვლას და მათში შესაბამისი ცვლილებების შეტანას. თავის მხრივ შეტყობინებები ამ ცვლილებების შესახებ აგრეთვე ვრცელდებიან ქსელში. ამგვარად დინამიური მარშრუტიზაცია ავსებს სტატიკურ მარშრუტიზაციას, სადაც ეს უკანასკნელი არ გამოდგება.

<u>ერთმარშრუტიანი და მრავალმარშრუტიანი.</u> მრავალმარშრუტიან ალგორითმებში შეიძლება არსებობდეს რამოდენიმე მარშრუტი ერთი დანიშნულების წერტილამდე და მოხდეს ამ წერტილში გადასაცემი მონაცემების მულტიპლექსირება ამ მარშრუტებით. ერთმარშრუტიანი ალგორითმები გულისხმობენ დროის ყოველ მომენტში მხოლოდ ერთი ასეთი მარშრუტის არსებობას. ცხადია მრავალმარშრუტიანი ალგორითმები გამოირჩევიან უფრო მაღალი გამტარუნარიანობით და საიმედოობით.

<u>ერთდონიანი და იერარქიული.</u> ერთდონიან ალგორითმებში ყველა მარშრუტიზატორი ერთმანეთის მიმართ ტოლი რანგისა არიან. იერარქიულ ალგორითმებში მონაცემები პერიფერიულ მარშრუტიზატორებიდან გადიან ბაზურ მარშრუტიზატორბზე, რომლებიც შეადგენენ მარშრუტიზაციის საფუძველს და გადაადგილდებიან, სანამ არ მიაღწევენ დანიშნულების ადგილის მიდამოს. აქ მონაცემები კვლავ გადაეცემა პერიფერიულ მარშრუტიზატორებს, რომლებიც გადასცემენ მათ დანიშნულების წერტილში.

<u>მარშრუტიზაცია წყაროდან.</u> ამ ტიპის ალგორითმებში მარშრუტიზატორები ახდენენ მხოლოდ მონაცემთა პაკეტების კომუტაციას, ე.ი. გადააგზავნიან მათ შემდეგ მარშრუტიზატორზე. ქსელში პაკეტის მარშრუტს კი ადგენს თვით მონაცემთა წყარო წინასწარ. დანარჩენ ალგორითმებში წყარომ არაფერი იცის ქსელის სტრუქტურის შესახებ. მარშრუტს ადგენენ მარშრუტიზატორები საკუთარი გათვლების თანახმად.

<u>შიდადომენური და დომენთაშორისი.</u> ზოგი ალგორითმები მოქმედებენ მხოლოდ დომენის შიგნით, სხვებს კი შეუძლიათ მოქმედება როგორც დომენის შიგნით ასე მის გარეთაც. დომენი წარმოადგენს ერთიანი ადმინისტრირების ქვეშ მყოფ ქსელის ნაწილს. დომენის შიდა სტრუქტურა გარე სამყაროსათვის უცნობია და ის ამგვარად წარმოადგენს დომენთაშორისი მარშრუტიზაციის ერთეულს. დომენთაშორისი ალგორითმები ძირითადად ემსახურებიან მარშრუტთა გაცვლას დომენებს შორის.

<u>არხის მდგომარეობის და დისტანციურ-ვექტორული ალგორითმები.</u> არხის მდგომარეობის ალგორითმებში სამარშრუტო ინფორმაცია გადაიგზავნება ქსელის ყველა კვანძში, მაგრამ ამ ინფორმაციაში შედის მხოლოდ მარშრუტიზატორის საკუთარი არხების მდგომარეობა. ხოლო დისტანციურ-ვექტორულ ალგორითმებში გადაიგზავნება მთლიანი სამარშრუტო ტაბულა ან მისი ნაწილი, მაგრამ მხოლოდ მოცემული მარშრუტიზატორის მეზობლებთან. გარდა ამისა, არხის მდგომარეობის ალგორითმები ითვალისწინებენ მარშრუტის შემადგენელი არხების მახასიათებლებს, როცა დისტანციურ ვექტორული ალგორითმები ითვალისწინებენ მარშრუტის სიგრძეს. არხის მდგომარეობის ალგორითმები გამოირჩევიან სწრაფი კრებადობით და ნაკლებად ქმნიან სამარშრუტო მარყუჟებს. მაგრამ ამ ტიპის ალგორითმები გაცილებით რთულია, ვიდრე დისტანციურ-ვექტორული ალგორითმები და მოითხოვენ მეტ პროცესორულ სიმძლავრეს და მეტ მეხსიერების მოცულობას.

სტატიკური მარშრუტიზაციის ალგორითმის ტრივიალობის გამო მოცემულ ნაშრომში განიხილება მხოლოდ დინამიური მარშრუტიზაციის ალგორითმები.

2. მარშრუტიზაციის ალგორითმები

მარშრუტიზაციის ალგორითმების ცნების ქვეშ ძირითადად იგულისხმება ქსელური დონის პროტოკოლები, რომელებიც მართავენ პაკეტების გადაადგილებას ქვექსელებში დანიშნულების ადგილამდე. ქსელის კვანძებში მარშრუტის არჩევისათვის გამოიყენება საკმაოდ რთული ალგორითმების ნაკრები, რომლებიც ახდენენ ერთმანეთში მონაცემთა გაცვლას. ეს სირთულე განპირობებულია იმ ფაქტით, რომ მოცემული ამოცანის გადაწყვეტა ითხოვს ქსელის ყველა კვანძის შეთანხმებულ მუშაობას. მარშრუტიზაციის სისტემა უნდა რეაგირებდეს კვანძებისა და ხაზების მწყობრიდან გამოსვლაზე ან გადატვირტვაზე და მიმართავდეს მონაცემებს სხვა მარშრუტით.

მოცემული ამოცანა შეიძლება პირობითად დაიყოს ორ ქვეამოცანად: 1) მარშრუტის არჩევა, რომელიც იძლევა საუკეთესო მაჩვენებლებს. აქ განიხილება უმოკლესი გზის პოვნის ალგორითმები და უფრო რთული ალგორითმი, რომელიც იძლევა მონაცემთა ნაკადის ოპტიმალურთან მიახლოვებულ მაჩვენებლებს; 2) მარშრუტის არჩევისათვის საჭირო ინფორმაციის გავრცელება ქსელის კვანძებს შორის.

მარშრუტის სწორი არჩევა მეტად მნიშვნელოვანი ამოცანაა, რადგან ეს საშუალებას იძლევა საკმაოდ გაიზარდოს ქსელის ჯამური გამტარუნარიანობა და მცირდება საშუალო დაყოვნება. მარშრუტიზაციის მეთოდები იყენებენ რიგ გრაფულ ალგორითმებს, რომლებიც შემდგომ იქნება მოყვანილი.

2.1 უმოკლესი გზის ალგორითმები

უმოკლესი გზის პოვნის ალგორითმები ემყარებიან იმ ფაქტს, რომ ქსელის ყოველ ხაზს ან სეგმენტს გააჩნია საკუთარი ფასი ანუ სიგრძე. უმოკლესი გზის პოვნის ამოცანა მდგომარეობს ქსელის ორ წერტილს შორის მოიძებნოს ისეთი მარშრუტი, რომელსაც გააჩნია მინიმალური ფასი. თუ ეს ფასი გამოხატავს ხაზის ფიზიკურ მახასიათებლებს, მაგალითად მის დატვირთულობის ხარისხს, მაშინ უმოკლესი ფასის მქონე გზის პოვნა გადაიქცევა ნაკლებად დატვირთული მარშრუტის პოვნის ამოცანად.

მარშრუტიზაციის ალგორითმებში ქსელის წარმოდგენილია ორიენტირებული $G=(N,A)$ გრაფის საშუალებით, სადაც N არის ქსელის კვანძების სიმრავლე, ხოლო A - ამ კვანძების შემაერთებელი ხაზების სიმრავლე. G გრაფის ყოველ $(i,j)\in A$ რკალს მიეწერება რაღაც d_{ij} ფასი. მაშინ ამ გარფში ნებისმიერი $p=(i,j,k,\ldots,l,m)$ ორიენტირებული გზის სიგრძე განისაზღვრება როგორც $d_{ij}+d_{jk}+\ldots d_{lm}$.. ამოცანა მდგომარეობს იმაში, რომ ნებისმიერი i და m კვანძთა წყვილისათვის მოიძებნოს ისეთი გზა, რომლის სიგრძე იქნება მინიმალური. გზის სიგრძე შეიძლება გამოხატავდეს ნებისმიერ კრიტერიუმს ან კრიტერიუმთა კომბინაციას, ამიტომ მოყვანილი ამოცანა გამოსადეგია არა მარტო გამოთვლით ტექნიკაში, არამედ სხვა დარგებშიც.

ახლა განვიხილოთ უმოკლესი გზის პოვნის რამოდენიმე ალგორითმი.

ბელმან-ფორდის ალგორითმი

დავუშვათ, რომ კვანძი 1 წარმოადგენს წყაროს და საჭიროა მოიძებნოს უმოკლესი გზების სიგრძეები გრაფის ყოველ კვანძამდე. დავუშვათ, რომ გრაფის თითოეული რკალის სიგრძე დადებითი რიცხვია. (სინამდვილეში ეს შეზღუდვა ნაკლებად მკაცრია: დასაშვებია რკალების როგორც დადებითი, ისე უარყოფითი სიგრძეები, ოღონდ გრაფში არ უნდა იყოს უარყოფითი სიგრძის ციკლები, რადგან ამ შემთხვევაში ალგორითმი ამ ციკლის ყოველ ჯერზე გავლით ამცირებს გზის სიგრძეს და არასოდეს არ დაასრულებს თავის მუშაობას). ეს შეზღუდვა ძირითადად

სრულდება კომპიუტერულ ქსელებში. აღვნიშნოთ $d_{ij}=\infty$, თუ გრაფში არ არსებობს რკალი (i,j). ბელმან-ფორდის ალგორითმის არსი იმაში მდგომარეობს, რომ ჯერ მოიძებნება უმოკლესი გზები იმ პირობით, რომ ისინი არ შეიცავენ არა უმეტეს 1 რკალისა, შემდგომ მოიძებნება არაუმეტეს 2 რკალის შემცველი გზები და ა.შ. უმოკლეს გზას, რომელიც შეიცავს არა უმეტეს h რკალისა ვუწოდით $(\le h)$ გზა, ხოლო ამ გზის შესაბამისი სიგრძე 1 კვანძიდან i კვანძამდე აღვნიშნოთ $D^{(h)}{}_i$-ით. ბელმან-ფორდის ალგორითმი შეიძლება შემდეგნაირად გამოისახოს:

დასაწყისში $D^{(0)}{}_i=\infty$, ყველა $i\neq 1$-ისათვის; ხოლო ყველა შემდგომი $h\ge 0$-ისათვის

$$D^{(h+1)}{}_i=\min_j[D^{(h)}{}_j+d_{ji}] \text{ , ყველა } i\neq 1\text{-ისათვის } (1)$$

ადვილი შესამჩნევია, რომ სეგმენტთა რაოდენობა უმოკლეს გზაში 1-დან i-მდე ვერ იქნება $N-1$-ზე მეტი, სადაც N კვანძთა რაოდენობაა (თუ გრაფში არ არის უარყოფითი ციკლები) და ტოლია $D^{(N-1)}{}_i$-ის. ამგვარად იტერაციების რაოდენობა $N-1$-ს ვერ აღემატება. გარდა ამისა, თუ $D^{(h+1)}{}_i=D^{(h)}{}_i$ ყველა i-სა და გარკვეული h-სათვის, მაშინ შემდგომი იტერაციები აღარ შეცვლიან უმოკლესი გზების მნიშვნელობებს და ამაზე ალგორითმის მუშაობა შეიძლება დასრულდეს და უმოკლეს გზების მნიშვნელობებად მიჩნეულ იქნეს $D^{(h)}{}_i$.

თუ აღვნიშნავთ უმოკლეს გზის სიგრძეს 1 კვანძიდან i კვანძამდე D_i-ით, მაშინ (1) წესი საბოლოოდ შეიძლება შემდეგნაირად გადავწეროთ:

$$D_i=\min_j[D_j+d_{ji}] \text{ , ყველა } i\neq 1\text{-სათვის,} \quad (2)$$

$$D_1=0.$$

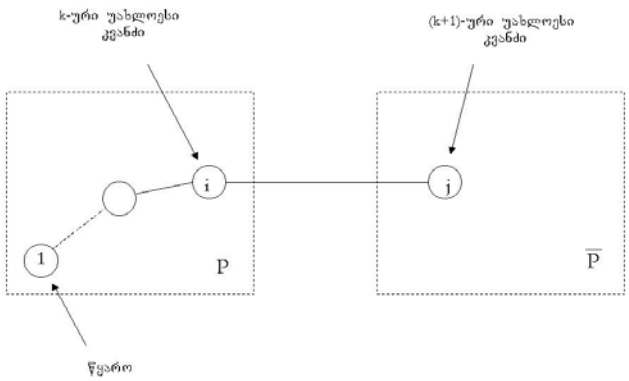
(2) არის ბელმანის განტოლება, რომლის ამოხსნა იძლევა უმოკლეს გზებს. მოცემული ალგორითმში ჯერ განიხილავენ i-ურ კვანძს და გადაადგილდებიან უკუ მიმართულებით, სანამ არ მიაღწევენ 1 კვანძს. ამის გამო ამ ალგორითმს აგრეთვე უწოდებენ უკუ ძებნის ალგორითმს (Backward Search Algorithm).

არსებობს მოცემული ალგორითმის განაწილებული რეალიზაცია, რომელიც ფართოდაა გამოყენებული TCP/IP ქსელებში. მას ჩვენ შემდგომ განვიხილავთ.

დიკსტრას ალგორითმი

ეს ალგორითმი მკაცრად მოითხოვს, რომ რკალების სიგრძეები იყოს დადებითი. დიკსტრას ალგორითმის არსი იმაშია, რომ კვანძებამდე უმოკლესი გზები მოიძებნება ამ გზების ზრდადობის მიმდევრობით. ყველა უმოკლეს გზას შორის უმცირესი ცხადია იქნება ერთი რკალისაგან შემდგარი გზა, რომელიც აერთებს 1 კვანძს მის უახლოვეს მეზობელთან. შემდეგი უმოკლესი გზა იქნება ან ერთრკალიანი გზა შემდეგ უახლოვეს მეზობლამდე, ან ორრკალიანი გზა, რომელიც გადის პირველ ბიჯზე არჩეულ კვანძზე.

ყოველ i კვანძს შევუსაბამოთ D_i უმოკლესი მანძილი 1 კვანძიდან i კვანძამდე. იმ კვანძების სიმრავლეს, რომელთა შესაბამისი D_i უკვე დადგენილია აღვნიშნოთ P-თი. მაშინ დიკსტრას ალგორითმი მდგომარეობს შემდეგში:



დასაწყისში $P=\{1\}$, $D_1=0$ და $D_j=d_{1j}$ ყველა $j\neq 1$-სათვის. შემდეგ მოიძებნება შემდგომი უახლოვესი კვანძი, ე.ი. მოინახოს ისეთი $i\notin P$, რომ

$$D_i = \min_{j\notin P} D_j$$

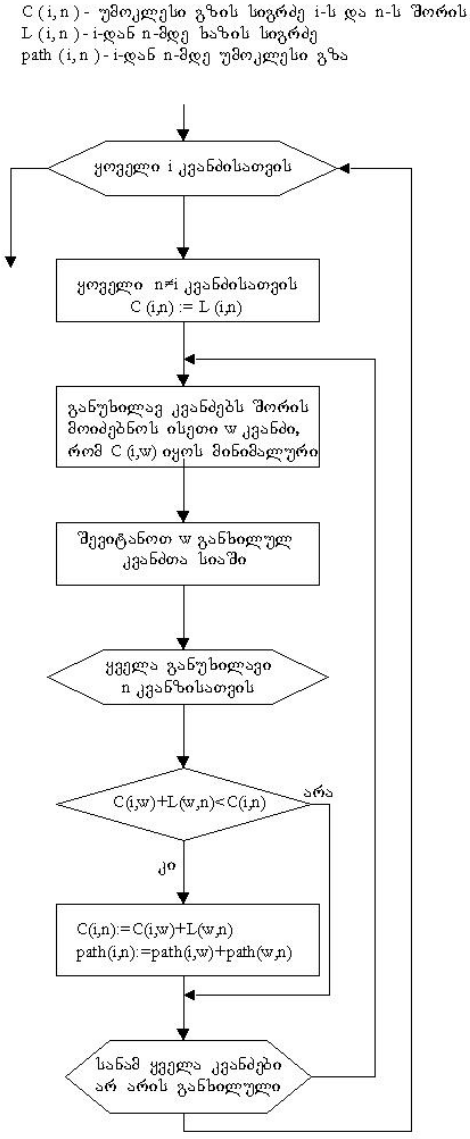
შევიტანოთ i კვანძი P სიმრავლეში: $P=P\cup\{i\}$.

ამის შემდეგ განვაახლოთ დარჩენილი კვანძების D პარამეტრები შემდეგი ფორმულის მიხედვით:

$$D_j=\min[D_j, D_i+d_{ij}].$$

რის შემდეგაც მოიძებნება შემდეგი უახლოვესი კვანძი i. ალგორითმი დაასრულებს თავის მუშაობას, როდესაც ყველა კვანძი ამოწურულია.

მოვიყვანოთ დიკსტრას ალგორითმის ბლოკ-სქემა:



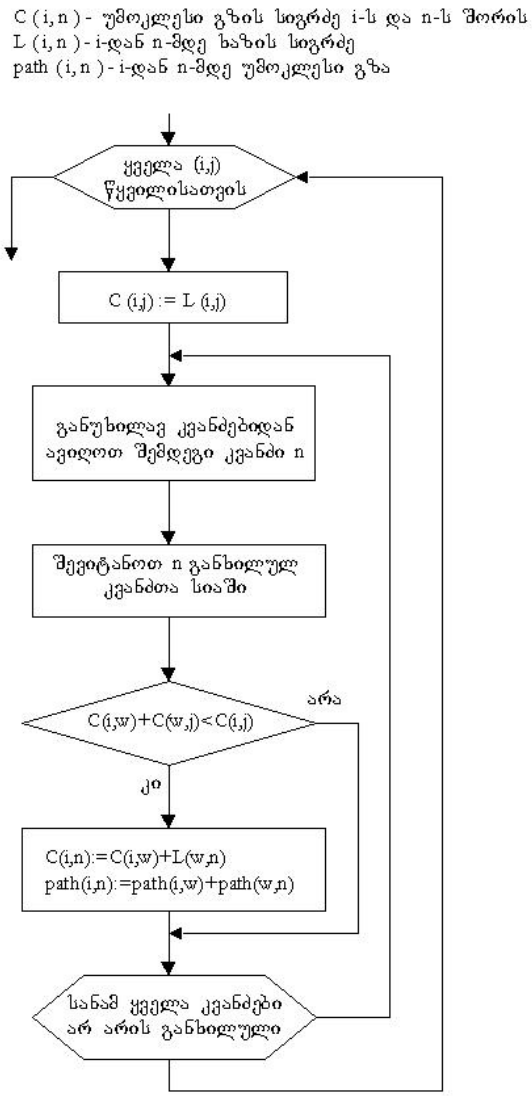
ფლოიდ-უორშელის ალგორითმი

ფლოიდ-უორშელის ალგორითმი წარმოადგენს დიკსტრას ალგორითმის ნაირსახეობას. ეს ალგორითმი პოულობს უმოკლეს გზებს ქსელის კვანძთა ყველა წყვილისათვის. აქ რკალების სიგრძეებისადმი წაყენებული მოთხოვნები ანალოგიურია ბელმან-ფორდის ალგორითმისა. მისი არსი შემდეგია: ნებისმიერ ორ კვანძს შორის განიხილება ჯერ პირდაპირი გზა (ე.ი. რომელიც არ შეიცავს შუალედურ კვანძებს), შემდეგ გზა, რომელშიც შუალედური კვანძი შეიძლება იყოს კვანძი 1, შემდეგ გზა, სადაც დასაშვები შუალედური კვანძებია 1 და 2, და ა.შ. $D^{(n)}_{ij}$-თი აღვნიშნოთ უმოკლესი გზა i და j კვანძებს შორის იმ პირობით, რომ მხოლოდ $1,2,...,n$ კვანძები შეიძლება გამოყენებულ იქნან, როგორც საშუალედო კვანძები. ახლა ჩამოვაყალიბოთ ალგორითმი:

საწყისი პირობებია: $D^{(0)}_{ij}=d_{ij}$ ყველა i და j-სათვის $i\neq j$. შემდეგ ყოველი $n=0,1,...,N-1$ -სათვის

$D^{(n+1)}_{ij}=\min[D^{(n)}_{ij}, D^{(n)}_{i(n+1)}+D^{(n)}_{(n+1)j}]$, ყველა $i\neq j$-სათვის.

განვიხილოთ ალგორითმის ბლოკ-სქემა:



ბელმან-ფორდის განაწილებული ასინქრონული ალგორითმი

განვიხილოთ ბელმან-ფორდის ალგორითმის რეალიზაცია, რომელიც შეიძლება შესრულდეს ქსელის კვანძებში ერთმანეთისაგან დამოუკიდებლად. ეს ალგორითმი ფართოდაა გამოყენებული TCP/IP ქსელებში.

დავუშვათ, რომ d_{ij} დადებითია ყოველი (i,j) ხაზისათვის. ამ ალგორითმში ერთ-ერთი ყველაზე მნიშვნელოვანი სიდიდეა D_i, რომელიც წარმოადგენს გზის სიგრძეს i-ური საწყისი კვანძიდან საერთო დანიშნულების კვანძამდე, მაგალითად კვანძი 1. (პრაქტიკაში ალგორითმი სრულდება ყველა შესაძლო დანიშნულების კვანძისათვის ცალკ-ცალკე). ამ შემთხვევისათვის ბელმანის განტოლებას ექნება შემდეგი სახე:

$$D_i = \min_{j \in N(i)} [d_{ij} + D_j], \quad i \neq 1,$$

$D_1=0,$

სადაც $N(i)$ აღნიშნავს მოცემულ მომენტში i კვანძის მეზობელთა სიმრავლეს, ე.ი. იმ კვანზთა სიმრავლეს, რომლებიც უშუალოდ არიან ხაზით დაკავშირებულნი i კვანძთან. შესაბამისად ამ შემთხვევისათვის ალგორითმის იტერაცია შეიძლება გამოიხატოს განტოლებით:

$$D^{(h+1)}{}_i = \min_{j\in N(i)}[d_{ij}+D^{(h)}{}_j],\ i\neq 1,$$

$$D^{(h+1)}{}_1=0.$$

მოცემული იტერაციები სრულდება ქსელის ყოველ კვანძზე დამოუკიდებლად და მისი შედეგები მიმოიცვლება კვანძებს შორის. ამ ალგორითმის შესრულების შედეგად ყოველმა i კვანძმა იცის არა მარტო D_i უმოკლესი მანძილი რომელიღაც დანიშნულების კვანძამდე, არამედ მისგან გამავალი ხაზი, რომელზეც ძევს ეს უმოლკესი გზა.

შემოვიღოთ აღნიშვნები:

$D^i{}_j(t)$- 1 კვანძამდე უმოკლესი გზის შეფასება ყოველი მეზობელი კვანძისათვის, რომელიც გადაეცა i კვანძს ბოლოჯერ.

$D_i(t)$- 1 კვანძამდე უმოკლესი გზის შეფასება i კვანძისათვის გამოთვლილი ბელმან-ფორდის ალგორითმის იტერაციის შესაბამისად ბოლოჯერ.

ზემოაღნიშნულის მიხედვით ყოველ კვანძში ალგორითმის მუშაობის პროცესში დროის ყოველ t მომენტში შეიძლება ადგილი ჰქონდეთ შემდეგ მოვლენებს:

1. კვანძი i ანახლებს $D_i(t)$-ს ფორმულით

$$D_i(t):= \min_{j\in N(i)}[d_{ij}+D^i{}_j(t)]$$

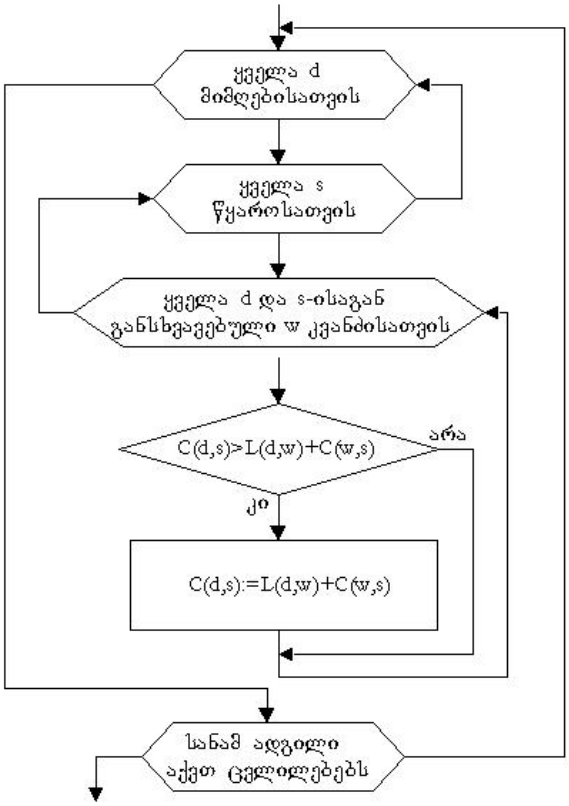
და ტოვებს $D^i{}_j(t),\ j\in N(i)$ შეფასებებს უცვლელად.

2. კვანძი i იღებს ერთი ან რამოდენიმე j მეზობლისაგან, $j\in N(i)$, D_j მნიშვნელობებს გამოთვლილს რომელიღაც $t_0<t$ დროის მომენტში, ანახლებს $D^i{}_j(t)$ შეფასებებს და ტოვებს დანარჩენ შეფასებებს უცვლელად.

3. კვანძი i თავისუფალია და ამ შემთხვევაში ყველა არსებული შეფასება უცველელი რჩება.

მოვიყვენოთ ბელმან-ფორდის ალგორითმის ბლოკ-სქემა გამარტივებული სახით, იმ პირობით, რომ ყოველ კვანძმა სრულად იცის მთელი ქსელის ტოპოლოგია და ნებისმიერი ხაზის სიგრძე:

C(i,n) - უმოკლესი გზის სიგრძე i-ს და n-ს შორის
L(i,n) - i-დან n-მდე ხაზის სიგრძე
path(i,n) - i-დან n-მდე უმოკლესი გზა



2.2 სამარშრუტო ინფორმაციის გავრცელების მეთოდები

კიდევ ერთი ამოცანა, რომელიც წამოიჭრება მარშრუტიზაციის პროცესში, ეს არის სამარშრუტო ინფორმაციის მიწოდება იმ ადგილებიდან, სადაც ის გამოითვლება იქ, სადაც ის საჭიროა. ამ ამოცანას ართულებს ის გარემოება, რომ ხაზები, რომლებითაც გადაიცემა ეს ინფორმაცია შეიძლება გამოიდიოდნენ მწყობრიდან.

მოცემული ამოცანის გადაწყვეტა დაკავშირებულია მთელ რიგ სიძნელეებთან, რომლებიც უნდა იქნენ გათვალისწინებულნი შესაბამის ალგორითმში. ჩამოვთვალოთ ისინი:

1. სამარშრუტო ინფორმაცია შეიძლება დაკარგული ან დამახინჯებული იქნას ქსელის რომელიმე ხაზის მწყობრიდან გამოსვლის გამო. ზოგიერთ შემთხვევაში საჭიროა ქსელის ფუნქციონირების შენარჩუნება მისი ბმულობის დაკარგვის შემთხვევაშიც კი.

2. მარავალჯერადი ტოპოლოგიური ცვლილებების შემთხვევაში გენერირდება სამარშრუტო შეტყობინებების მთელი სიმრავლე. მაგრამ ამ ინფორმაციის გავრცელების პროცესში წამოიჭრება ახალი და მოძველებული ინფორმაციის განსხვავების საჭიროება.

3. სამარშრუტო ინფორმაცია გამოიყენება რაიმე ალგორითმის მიერ. თუ ეს ინფორმაცია მოვიდა კვანძში ალგორითმის მუშაობის პროცესში, ამ უკანასკნელმა უნდა ან დინამიურად გაითვალისწინოს იგი, ან დაიწყოს თავისი მუშაობა თავიდან.

4. რომელიმე ხაზის ფუნქციონირების აღდგენის შემთხვევაში ქსელის ორი არაბმული ნაწილი შეიძლება კვლავ გახდეს ბმული. ამ ნაწილებიდან ყოველი შესაძლოა შეიცავდეს მოძველებულ ინფორმაციას მეორე ნაწილის ტოპოლოგიის შესახებ. ამ შემთხვევაში ორივე ნაწილი სასრულ დროში უნდა შეთანხმდენ ქსელის ჭეშმარიტ ტოპოლოგიაზე.

ახლა მოვიყვანოთ სამარშრუტო ინფორმაციის განაწილების რამოდენიმე ალგორითმის აღწერა.

ზვავური ალგორითმი

ეს ალგორითმი ფართოდაა გავრცელებული TCP/IP ქსელებში და დიდი პოპულარობა მოიპოვა თავისი შედარებითი სიმარტივის გამო. ამ ალგორითმის შინაარსი შემდეგში მდგომარეობს: რომელიმე ხაზის სტატუსის ან მახასითებლების ცვლილების შემთხვევაში მასთან უშუალოდ დაკავშირებული კვანძები გადასცემენ შეტყობინებას ამის შესახებ თავიანთ მეზობელ კვანძებს. ეს უკანასკნელნი კი თავის მხრივ გადასცემენ თავიანთ მეზობლებს გარდა იმ კვანძისა, რომლისგანაც მათ მიიღეს ეს შეტყობინება. ამგვარად ინფორმაცია ქსელის მდგომარეობის ცვლილების შესახებ ვრცელდება ქსელში, მაგრამ ამ შემთხვევაში შესაძლოა გაჩნდნენ პრობლემები ქსელში ციკლების არსებობისას..

ციკლის არსებობისას მასში შეტყობინების მოხვედრისას იგი იტრიალებს იქ უსასრულოდ. ამ პრობლემის თავიდან ასაცილებლად შემოღებულია ე.წ. რიგითი ნომრების სისტემა. ქსელში სამარშრუტო შეტყობინებები ყოველი კვანძისათვის აღინიშნება შესაბამისი რიგითი ნომრებით. თუ კვანძი `j` მიიღებს შეტყობინებას, რომელიც წარმოიქმნა კვანძში `i`, მაშინ ის გადასცემს ამ შეტყობინებას თავის მეზობლებს მხოლოდ იმ შემთხვევაში, თუ ამ შეტყობინების რიგითი ნომერი მეტია იგივე კვანძიდან მოსული წინა შეტყობინების რიგით ნომერზე. რიგითი ნომრის სიგრძე უნდა იყოს საკმარისად დიდი, რომ ადგილი არ ჰქონდეს გადავსებას. (მაგალითად 48 ბიტიანი ველი უზრუნველყოფს ალგორითმის მუშაობას 500 წლის განმავლობაში იმ პირობით, რომ შეტყობინებები გადაიცემა ყოველ მილიწამს).

დიდ პრობლემას წარმოადგენს ის ფაქტი, რომ რიგითი ნომრები შესაძლოა დამახინჯებულ იქნან კვანძის მეხსიერებაში შენახვის პროცესში. რიგითი ნომრის შემთხვევითმა გადიდებამ შეიძლება გამოიწვიოს ის, რომ `i` კვანძიდან მოსული შემდგომი შეტყობინებები აღარ მიიღება, სანამ ჭეშმარიტი რიგითი ნომერი არ გადააჭარბებს შეცდომითს. აგრეთვე პრობლემა წამოიჭრება იმ შემთხვევაში, როცა ქსელში დაკარგულია ბმულობა რაღაც დროის შუალედის განმავლობაში, რის შედეგადაც შეიძლება საჭირო გახდეს ზოგი რიგითი ნომრების განულება, რადგან შესაბამის კვანძს დაავიწყდა ბოლო რიგითი ნომერი. ამ პრობლემების გადასაჭრელად მიიღება შემდეგი ზომები:

1. ყოველი სამარშრუტო შეტყობინება შეიცავს ასაკის (TTL) ველს, რომელიც აღნიშნავს, თუ რამდენი ხანია დასაშვები მოცემული შეტყობინების არსებობა ქსელში. ნებისმიერ კვანძში მიღებისას ის გამოითვლის შეტყობინების გადაგზავნაზე დახარჯულ დროს და გამოაკლებს მას TTL სიდიდეს. როგორც კი TTL ამოიწურება, შეტყობინება შემდგომ აღარ გადაიცემა. მოძველებული შეტყობინება ყოველთვის გადაიფარება არამოძველებულით რიგითი ნომრის მიუხედავად, ხოლო არამოძველებულ შეტყობინებას გადაფარავს მხოლოდ უფრო მაღალი რიგითი ნომრის მქონე შეტყობინება. ეს გარანტიას იძლევა, რომ

შეცდომით გაზრდილი რიგითი ნომერი ძალაში იქნება შედარებით მცირე დროის განმავლობაში.

2. სამარშრუტო შეტყობინებები გადაიცემა არა მარტო ქსელში რომელიმე ხაზის სტატუსის ცვლილების შემთხვევაში, არამედ პერიოდულად მეორდება (დაახლოვებით 60 წმ-ში ერთხელ). ეს საშუალებას იძლევა აღდგეს სამარშრუტო ინფორმაცია ქსელში მისი ბმულობის აღდგენის შემთხვევაში.

შესაძლებელია ზვავური ალგორითმის განხორციელება ასაკის ველისა და შეტყობინებების პერიოდული გაგზავნის გარეშეც. ამისათვის მწყობრიდან გამოსული ხაზის აღდგენისას ამ ხაზის ბოლო კვანძებმა უნდა გაცვალონ ინფორმაცია ქსელის ტოპოლოგიის შესახებ. ამ მიზნით ისინი თავის ბოლო რიგით ნომერს უტოლებენ 0-ს და შემდეგ გაცვლიან ყველა მიღებულ სამარშრუტო შეტყობინებებს, რის შემდეგ აღდგება ინფორმაცია ტოპოლოგიის შესახებ და ყოველი კვანძი "გაიხსენებს" თავის უდიდეს რიგით ნომერს, რის შემდეგაც შეძლებს ახალი შეტყობინებების გაგზავნას და ამგვარად ქსელი გადავა ნორმალური ფუნქციონირების რეჟიმში.

შემოვიღოთ აღნიშვნა: $A>B$, თუ A-ს რიგითი ნომერი მეტია B-ს რიგით ნომერზე, ან თუ ისინი ტოლებია A-ს შიგთავსი მეტია B-ს შიგთავსზე რაღაც ლექსიკოგრაფიული წესის მიხედვით. ანალოგიურად განისაზღვრება $B>A$ და $A=B$.

დავუშვათ, რომ j კვანძმა მიიღო A შეტყობინება გენერირებული i კვანძში. შეტყობინება უგულვებელყოფილი იქნება, თუ $A \leq B$, ხოლო თუ $A>B$ მოიქცევა შემდეგი წესის შესაბამისად:

1. თუ $j \neq i$, მაშინ j ჩაწერს ამ შეტყობინებას თავის მეხსიერებაში და გააგზავნის მას თავისი ყველა გამავალი ხაზით, გარდა ინ ხაზისა, საიდანაც ეს შეტყობინება მოვიდა.

2. თუ $j=i$, ე.ი. კვანძმა მიიღო თავის მიერვე გენერირებული შეტყობინება, მაშინ ეს შეტყობინება იგნორირებულ იქნება და i კვანძი გააგზავნის ახალ შეტყობინებას თავისი ხაზების მდგომარეობის შესახებ, რომლის რიგითი ნომერი 1-ით აღემატება A შეტყობინების რიგით ნომერს.

იმისათვის, რომ გავიგოთ რისთვისაა საჭირო შეტყობინებების ასეთი შედარების ჩატარება, განვიხილოთ მაგალითი: დავუშვათ გვაქვს 3 კვანძიანი ქსელი. ეს კვანძები შეერთებულია $(1,2)$ და $(2,3)$ ხაზებით. ყველა კვანძი შეიცავს სწორ ინფორმაციის მქონე შეტყობინებებს, რომელთა რიგითი ნომერია 0. დავუშვათ, რომ მწყობრიდან გამოვიდა ჯერ $(2,3)$ ხაზი, ხოლო შემდეგ $(1,2)$ ხაზი. და ბოლოს $(2,3)$ ხაზი აღდგა. 2 და 3 კვანძები მიმოცვლიან საწინააღმდეგო ინფორმაციას $(1,2)$ ხაზის შესახებ. პირველად განხილული ზვავური ალგორითმის მოდიფიკაციის შემთხვევაში ეს შეტყობინებები უგულვებელყოფილი იქნებოდა, რადგან ახალ შეტყობინებებსაც და მეხსიერებაში არსებულ შეტყობინებებს აქვთ 0-ის ტოლი რიგითი ნომერი. ამ შემთხვევაში კი 2 კვანძში არსებული სწორი ინფორმაცია ან მაშინვე გამოაძევებს 3 კვანძში არსებულ მოძველებულ ინფორმაციას, ან ეს მოხდება 1-ის ტოლი რიგითი ნომრიანი შეტყობინების მიღებისას 3 კვანძში.

მოყვანილ ალგორითმს გააჩნია მნიშვნელოვანი ნაკლი, რომელიც ჩნდება შეტყობინებების არაპერიოდულობის შემთხვევაში. თუ რომელიმე კვანძის მეხსიერებაში შეტყობინების რიგითი ნომერი შეცდომით გაიზარდა, შემდგომ მოსული შეტყობინებები უგულვებელყოფილი იქნება, სანამ მათი რიგითი ნომერი არ გადააჭარბებს შეცდომით მიღებულ რიგით ნომერს.

მოცემული პრობლემა შეიძლება მოიხსნას, თუ ზემოთ მოყვანილ წესებს დაემატება კიდევ შემდეგი წესი: თუ j კვანძში მოვიდა A შეტყობინება და ამ კვანძის მეხსიერებაში ჩაწერილია B შეტყობინება ისეთი, რომ $A<B$, მაშინ A შეტყობინება, როგორც ადრე აღინიშნა, უგულვებელყოფილ იქნება, მაგრამ დამატებით B შეტყობინება გაიგზავნება იმ მეზობლის მიმართულებით, საიდანაც მოვიდა A. ეს მეზობელი შემდგომ გაავრცელებს B-ს ქსელში. ამგვარად, თუ გავრცელდა შეტყობინება, რომლის რიგითი ნომერი k ნაკლებია, ვიდრე რომელიმე კვანძის მეხსიერებაში არსებული შეტყობინების რიგითი ნომერი, მაშინ ამ კვანძის მიერ გაგზავნილი პასუხი ბოლოს მიაღწევს ამ შეტყობინების წყაროს, მაშინ ეს უკანასკნელი დააგენერირებს $(k+1)$ რიგითი ნომრის მქონე შეტყობინებას.

ალგორითმის ეს უკანასკნელი ვარიანტი შეტყობინებების პერიოდულობის გაუქმებით მნიშვნელოვნად ამცირებს მონაცემთა ნაკადების ინტენსივობას, თუმცა იწვევს დამატებით მონაცემთა ნაკადს უკუ მიმართულებით.

გავრცელების მეთოდი რიგითი ნომრების გარეშე

ამ ალგორითმს სხვაგვარად უწოდებენ უმოკლესი გზის ტოპოლოგიურ ალგორითმს. ეს ალგორითმი ზვავური ალგორითმისაგან განსხვავებით არ იყენებს რიგით ნომრებს და ამიტომ მას არ გააჩნია რიგითი ნომრების გადავსებასთან და დაკარგვასთან დაკავშირებული პრობლემები. უმოკლესი გზის ტოპოლოგიური ალგორითმი ეფუძნება შემდეგ პრინციპებს: როცა რომელიმე კვანძში მოდის საწინააღმდეგო ინფორმაცია, ის მიიღებს იმ ინფორმაციას, რომელსაც თვლის ყველზე სანდოდ. ინფორმაციის სანდოობა გამოიხატება მარშრუტის სიგრძით მის წყარომდე და განახლდება ახალი ინფორმაციის შემოსვლის მომენტში. ამგვარად ინფორმაციის სანდოობის მოძებნის პრობლემა მსგავსია უმოკლესი გზის პოვნის ამოცანის.

ყოველ კვანძში ინახება შემდეგი ინფორმაცია:

1. ძირითადი ტოპოლოგიური ტაბულა T_i. ამ ტაბულის მიხედვით ხდება გადაწყვეტილებების მიღება და მასში ინახება ის მარშრუტები კვანძებამდე, რომლებსაც კვანძი თვლის ყველაზე სანდოდ. მოცემული ალგორითმის მიზანია შეათანხმოს ყველა კვანძის ძირითადი სამარშრუტო ტაბულები.
2. საპორტო ტოპოლოგიური ტაბულები T^i_j. ასეთი ტაბულა არსებობს ყოველი მეზობელი j კვანძისათვის. ამ ტაბულაში ჩაიწერება მთელი ინფორმაცია ქსელის შესახებ, რომელსაც i კვანძი მიიღებს j მეზობელი კვანძისაგან.

ალგორითმი შეიძლება გამოისახოს შემდეგი 5 წესის საშუალებით:

ინფორმაციის გაცვლის წესები:

1. როგორც კი იცვლება ინფორმაცია რომელიმე ხაზის შესახებ ძირითად ტოპოლოგიურ ტაბულაში, მოცემული ჩანაწერი მაშინვე გადაიცემა ამ კვანძთან შეერთებული და მოფუნქციონირე ყველა ხაზით.
2. როცა მწყობრიდან გამოსული ხაზი ისევ დგება მწყობრში, მისი ბოლო კვანძები მაშინვე გაცვლიან თავიანთ ძირითად ტოპოლოგიურ ტაბულებს. ამის შემდეგ ეს კვანძები ჩაწერენ ამ კვანძის სტატუსს თავიანთ ძირითად და საპორტო ტოპოლოგიურ ტაბულებში.

ტოპოლოგიური ტაბულების განახლების წესები:

3.როგორც კი კვანძის მიმდებარე რომელიმე ხაზი გამოდის მწყობრიდან, ინფორმაცია ამის სესახებ მაშინვე შეიტანება ძირითად და საპორტო ტოპოლოგიურ ტაბულებში.

4.როდესაც კვანძი მიიღებს მეზობლისგან ინფორმაციას რომელიმე ხაზის მდრომარეობის შეცვლის შესახებ, მაშინვე შეაქვს ეს ინფორმაცია ამ მეზობლის შესაბამის საპორტო ტოპოლოგიურ ტაბულაში.

5.თუ შეიცვალა ჩანაწერი T_i ძირითად ტოპოლოგიურ ტაბულაში რომელიმე მიმდებარე ხაზის მდგომარეობის შეცვლასთან დაკავშირებით, ან T^i_j საპორტო ტოპოლოგიურ ტაბულაში j კვანძისაგან, i კვანძი ანახლებს თავის ძირითად ტოპოლოგიურ ტაბულას გარკვეული ალგორითმის მიხედვით:

ეს ალგორითმი ანალოგიურია დიკსტრას ალგორითმისა, იმ დაშვებით, რომ ყოველი მოქმედი ხაზის სიგრძე 1-ის ტოლია, ხოლო მწყობრიდან გამოსულისა კი - უსასრულობის. ყოველ k-ურ იტერაციაზე არსებობს კვანძთა P_k სიმრავლე, რომელიც შეიცავს იმ კვანძებს, რომლებიც შესაძლოა i კვანძიდან მიღწეულ იქნან არა უმეტეს k ხაზის გავლით. P_k სიმრავლის ყველა m კვანძს გააჩნია ჭდე, რომელიც აღნიშნავს i-ს იმ მეზობელი კვანძის ნომერს, რომელიც ძევს მოქმედი ხაზებისაგან შემდგარ უმოკლეს გზაზე i-სა და m -ს შორის. ყველა ისეთი ხაზის სტატუსი, რომლთა რომელიმე ბოლო შედის P_k სიმრავლეში, მაგრამ არც ერთი ბოლო არ შედის P_{k-1} სიმრავლეში, ე.ი. ხაზები სიმრავლიდან

$$L_k=\{(m,n)|m\notin P_{k-1},\ n\notin P_{k-1},\ (m\vee n)\in P_k\}$$

შეიტანება T_i ძირითად ტოპოლოგიურ ტაბულაში. დავუშვათ, რომ $P_0=\{i\}$ L_1-ის განსაზღვრისას. ყოველ k-ურ იტერაციაზე ხდება შემდეგი:

ყოველი $(m,n)\in L_k$ ხაზისათვის თუ ვთქვათ $m\in P_k$ და j არის მისი ჭდე, მაშინ ამ ხაზის სტატუსი კოპირდება შესაბამისი T^i_j საპორტო ტოპოლოგიური ტაბულიდან T_i ძირითად ტოპოლოგიურ ტაბულაში. თუ ეს ხაზი მოქმედია და $n\notin P_k$, მაშინ n-ს მივაწერთ ჭდეს j.

1.დავუშვათ M_k არის კვანძთა სიმრავლე, რომელთაც მიენიჭათ ჭდეები k-ურ იტერაციაზე. თუ M_k ცარიელია, ალგორითმი ასრულებს თავის მუშაობას. წინააღმდეგ შემთხვევაში $P_{k+1}=P_k\cup M_k$ და გადავიდეთ $(k+1)$-ურ იტერაციაზე.

მოცემული ალგორითმი შეიძლება გამოყენებულ იქნას არა მარტო ხაზების სტატუსების გადასაცემად, არამედ სხვა სამარშრუტო ინფორმაციის (დაყოვნების დროები, გამტარუნარიანობა და სხვა) გასავრცელებლადაც. მიუხედავად იმისა, რომ ეს ალგორითმი არ გამოიყენება TCP/IP ქსელებში, მისი მოყვანა აუცილებელი იყო იმის საილუსტრაციოდ, თუ როგორ შეიძლება დაძლეულ იქნას ზვავურ ალგორითმში არსებული პრობლემები.

2.3. ოპტიმალური მარშრუტიზაციის მეთოდები

ოპტიმალური მარშრუტიზაციის ცნება მჭიდროდაა დაკავშირებული ქსელის დატვირთულობის ცნებასთან. ცხადია, კარგი მარშრუტიზაციისას ქსელში პაკეტის დაყოვნების საშუალო სიდიდე და დისპერსია უნდა იყოს მინიმალური. მაგრამ ოპტიმიზაციისათვის მისაღები ერთიანი ფუნქციონალის მოძებნა მეტად ძნელია.

ხაზების დატვირთულობის შესაფასებლად ფართოდ გამოიყენება საშუალო ტრაფიკის სიდიდე, ე.ი. (i,j) ხაზის დატვირთულობის ქვეშ იგულისხმება შემომავალი ტრაფიკის ინტენსივობა F_{ij}, რომელსაც აგრეთვე უწოდებენ F_{ij} ნაკადს და რომელიც იზომება მონაცემთა ერთეულებით წამში (მაგალითად ბტ/წმ).

ოპტიმიზაციისას ღირებულებით ფუნქციად ხშირად ირჩევენ შემდეგი სახის გამოსახულებას:

$$\sum_{(i,j)} D_{ij}(F_{ij}), \qquad (2)$$

სადაც ყოველი D_{ij} მონოტონურად ზრდადი ფუნქციაა. D_{ij} ფუნქციად ხშირად გამოიყენება შემდეგი ფორმულა:

$$D_{ij}(F_{ij})=F_{ij}/(C_{ij}-F_{ij})+d_{ij}F_{ij},$$

სადაც C_{ij} არის ხაზის გამტარუნარიანობა, ხოლო d_{ij} არის ინფორმაციის გავრცელებითა და დამუშავებით გამოწვეული დაყოვნების სიდიდე. სხვა ფუნქციას, რომელიც შეიძლება გამოყენებულ იქნას ოპტიმიზაციის ღირებულებით ფუნქციაში, აქვს შემდეგი სახე:

$$D_{ij}(F_{ij})= \max_{(i,j)}\{F_{ij}/C_{ij}\}$$

ე.ი. ხაზთა გამოყენების კოეფიციენტის მაქსიმუმი.

ჩამოვაყალიბოთ ოპტიმალური მარშრუტიზაციის ამოცანა: ყოველი $w(i,j)$ (წყარო-მიმღები) წყვილისათვის პაკეტების შემოსვლის პროცესი ითვლება სტაციონალურად და გააჩნია ინტენსივობა r_w. ოპტიმალური მარშრუტიზაციის მიზანია r_w ინტენსივობის შემომავალი ტრაფიკი განაწილდეს რამოდენიმე მარშრუტზე წყაროსა და მიმღებს შორის ისე, რომ ქსელის ხაზებში წარმოქმნილი ჯამური ნაკადი ამინიმიზირებდეს (2) ღირებულებით ფუნქციას. შემოვიღოთ აღნიშვნები:

W - ყველა კვანძთა წყვილების სიმრავლე;

P_w - ყველა ორიენტირებული მარშრუტის სიმრავლე, რომლებიც აერთებენ w წყვილს წყაროს მიმღებთან;

r_p - ნაკადის სიდიდე p მარშრუტით ანუ სამარშრუტო ნაკადი.

ყველა $\{r_p|w\in W,\ p\in P_w\}$ სიდიდე უნდა აკმაყოფილებდნენ შემდეგ პირობებს:

$$\sum_{p\in P_w} x_p=r_w,\ \ w\in W,$$
$$x_p\geq 0,\ \ p\in P_w,\ \ w\in W. \qquad (3)$$

ამის გათვალისწინებით ღირებულებითი ფუნქცია მიიღებს შემდეგ სახეს:

$$\sum_{(i,j)} D_{ij}(F_{ij}) = \sum_{(i,j)} D_{ij}\Big[\sum_{(i,j)\in p} x_p\Big]$$

(3) შეზღუდვების გათვალისწინებით. მოცემული ამოცანის ამოხსნის შედეგი იქნება $x^*=\{r_p|w\in W,\ p\in P_w\}$ ოპტიმალური სამარშრუტო ნაკადთა ვექტორი.

მოცემული ამოცანის ამოხსნისათვის დავუშვათ, რომ D_{ij} ფუნქცია არის ორჯერ დიფერენცირებადი $[0,C_{ij})$ შუალდზე და მისი წარმოებულები D_{ij}' და D_{ij}'' მკაცრად დადებითი რიცხვებია ნებისმიერი F_{ij}-სათვის. გარდა ამისა, $D_{ij}(F_{ij})\to\infty$, როცა $F_{ij}\to C_{ij}$. მოცემული ამოცანის ამოხსნისათვის ჩამოვაყალიბოთ შემდეგი ოპტიმალურობის თეორემა:

თუ f არის დიფერენცირებადი, ამოზნექილი n განზომილებიანი $x=(x_1,\ldots,x_n)$ ვექტორის ფუნქცია და X - ვექტორთა ამოზნექილი სიმრავლეა. მაშინ $x^*\in X$ იქნება $f(x)$ ფუნქციის მინიმიზაციის ამოცანის ამონახსნი მაშინ და მხოლოდ მაშინ, როცა

n

$$\sum_{i=1} (\partial f(x^*)/\partial x_i)(x_i - x_i^*) \geq 0, \quad x \in X. \qquad (4)$$

ეს თეორემა ადვილად მტკიცდება, თუ განვიხილავთ ფუნქციას $g(\alpha)=f[x^*+\alpha(x-x^*)]$. ის $[0,1]$ შუალედზე აღწევს მინიმუმს, როცა $\alpha=0$, ე.ი. $dg(0)/d\alpha \geq 0$, ხოლო რთული ფუნქციის დიფერენცირების წესიდან გამომდინარე

$$dg(0)/d\alpha = \sum_{i=1}^{n} (\partial f(x^*)/\partial x_i)(x_i - x_i^*)$$

რაც ამტკიცებს ზემოთ მოყვანილ თეორემას.

დასმული ამოცანის ამოხსნა მოგვცემს ქსელის სამარშრუტო ნაკადთა ვექტორს x,

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

$D(x)$-ით აღვნიშნოთ ოპტიმიზაციის ღირებულებითი ფუნქცია.

$$D(x) = \sum_{(i,j)} D_{ij}\left[\sum_{(i,j)\in p} x_p\right], \qquad (5)$$

შესაბამისად

$$\partial D(x)/\partial x_p = \sum_{(i,j)\in p} D'_{ij}.$$

აქედან ჩანს, რომ $\partial D(x)/\partial x_p$ იქნება მთელი გზის სიგრძე, თუ D'_{ij} პირველი რიგის წარმოებულს მივიღებთ ყოველი (i,j) ხაზის სიგრძედ. ამიტომ $\partial D(x)/\partial x_p$ სიდიდეს უწოდებენ p გზის პირველწარმოებულ სიგრძეს.

თუ შევაჯამებთ (3), (4) და (5) გამოსახულებებს, მივიღებთ

$$\sum_{w\in W} \sum_{p\in P_w} (\partial D(x^*)/\partial x_p)(x_p - x_p^*) \geq 0, \quad x_p \geq 0,$$

და $p \in P_w$ ისეთი, რომ $\sum_{p\in P_w} x_p = r_w, \quad w \in W.$

ეს პირობა შემდეგი პირობის ექვივალენტურია:

$\partial D(x^*)/\partial x_{p'} \geq \partial D(x^*)/\partial x_p$, ყველა $p' \in P_w$ -სათვის. (6)

ამ პირობის არსი იმაში მდგომარეობს, რომ სამარშრუტო ნაკადთა ნაკრები ოპტიმალურია მხოლოდ და მხოლოდ მაშინ, როცა სამარშრუტო ნაკადი დადებითია მარტო იმ გზებისათვის, რომელთა პირველწარმოებული სიგრძე მინიმალურია. აქედან გამომდინარეობს, რომ იმ ხაზების პირველწარმოებული სიგრძეები, რომლებზედაც გაივლის r_w ტრაფიკის ნაწილი, ერთმანეთის ტოლია.

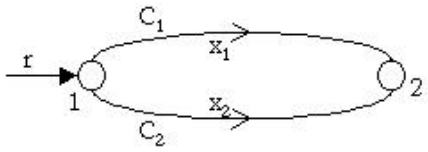
როგორც ადრე აღინიშნა, მოცემული მსჯელობა ჭეშმარიტია მხოლოდ იმ დაშვებით, რომ შემომავალ ნაკადთა ინტენსივობა არ იცვლება დროში. დროში ცვლად შემომავალ ნაკადთა ინტენსივობის შემთხვევაში ამოცანის ამოხსნისას შეგვიძლია გამოვიყენოთ არა თვით სამარშრუტო ნაკადთა ინტენსივობის მნიშვნელობები, არამედ მათი წილები კვანძში შემომავალი ნაკადის ინტენსივობაში

$\xi_p = x_p^*/r_w$, ყველა $p \in P_w$ -სათვის

და მოვითხოვოთ, რომ ყოველი w კვანძთა წყვილი ყოფდეს ტრაფიკს ამოცანის ამოხსნის შედეგად მიღებული პროპორციის შესაბამისად.

მოყვანილი მასალის საილუსტრაციოდ განვიხილოთ მარტივი მაგალითი:

განვიხილოთ მარტივი ქსელი, სადაც კვანძი 1 (წყარო) და კვანძი 2 (მიმღები) დაკავშირებულია ორი ხაზის საშუალებით.



საჭიროა 1 კვანძში შემომავალი წინასწარ ცნობილი r ნაკადი დაიყოს ორ სამარშრუტო ნაკადად x_1 და x_2 ისე, რომ მოხდეს ღირებულებითი ფუნქციის მინიმიზაცია:

$$D(x)=D_1(x_1)+D_2(x_2),$$

სადაც $D_i(x_i)=x_i/(C_i-x_i)$, აქ C_i არის i-ური ხაზის გამტარუნარიანობა. ამოცანას აზრი რომ ჰქონდეს, იგულისხმება, რომ r ნაკლებია C_1+C_2 ქსელის მაქსიმალურ გამტარუნარიანობაზე. (3) პირობის თანახმად, $x_1^*+x_2^*=r$, $x_1^*\geq 0$, $x_2^*\geq 0$. დავუშვათ, რომ $C_1\geq C_2$, მაშინ აშკარად $x_1\geq x_2$. ამ შემთხვევაში შესაძლებელია მხოლოდ ორი შემთხვევა:

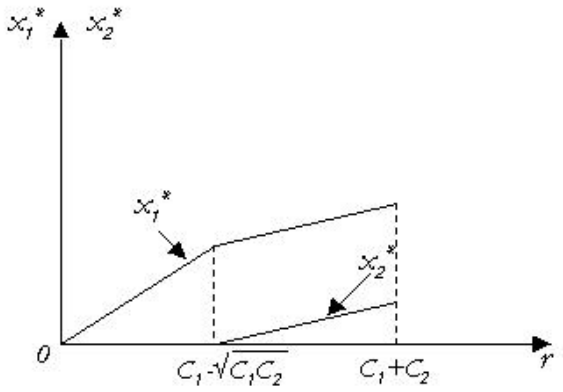
1. $x_1^*=r$ და $x_2^*=0$. მაშინ (6) თანახმად
   $\partial D_1(r)/\partial x_1 \leq \partial D_2(0)/\partial x_2$, აქედან გაწარმოების შედაგად მივიღებთ:
   $C_1/(C_1-r)\leq 1/C_2$,
   $r\leq C_1-\sqrt{C_1C_2}$.
2. $x_1^*\geq 0$ და $x_2^*\geq 0$. ამ შემთხვევაში ხაზების პირველწარმოებული სიგრძეები ტოლია, ამიტომ
   $\partial D_1(x_1)/\partial x_1 = \partial D_2(x_2)/\partial x_2$
   ეს ექვივალენტურია შემდეგი განტოლების:
   $C_1/(C_1-x_1^*)^2=C_2/(C_2-x_2^*)^2$.

თუ გავითვალისწინებთ, რომ $x_1^*+x_2^*=r$, ამონახსნი შედეგი იქნება:

$$x_1^*=\frac{\sqrt{C_1}\,[r-(C_2-\sqrt{C_1C_2})]}{\sqrt{C_1}+\sqrt{C_2}}$$

$$x_2^*=\frac{\sqrt{C_2}\,[r-(C_1-\sqrt{C_1C_2})]}{\sqrt{C_1}+\sqrt{C_2}}$$

მოცემული ამონახსნის მიხედვით განვიხილოთ ორივე ხაზის დატვირთულობის გრაფიკები.



აქ ჩანს, რომ r იღებს მნიშვნელობებს $[0, C_1+C_2)$ შესაძლო შუალედიდან. საკმაოდ მცირე r-ისათვის მთელი მონაცემთა ნაკადი მიემართება უფრო დიდი გამტარუნარიანობის მქონე ხაზში, ხოლო $C_1 - \sqrt{C_1 C_2}$ ზღვრის შემდეგ უკვე იყოფა ორ ხაზს შორის და r-ის შემდგომ ზრდასთან ერთად ამ ხაზებში ნაკადები იზრდებიან ისეთნაირად, რომ მათი პირველ-წარმოებული სიგრძეები ტოლი დარჩეს.

ნაკადთა დევიაციის ფრენკ-ვოლფის მეთოდი

არსებობს ოპტიმალური მარშრუტიზაციის რამოდენიმე ალგორითმი. ერთ-ერთი მათგანია ნაკადთა დევიაციის ფრენკ-ვოლფის მეთოდი, რომელიც განეკუთვნება ე.წ. დასაშვები მიმართულების ალგორითმების ჯგუფს. ამ ალგორითმის ძირითადი არსი იმაში მდგომარეობს, რომ მონაცემთა ნაკადის ნაწილი გადაიყვანება უმცირესი პირველწარმოებული სიგრძის მქონე გზებზე არამინიმალური სიგრძის მქონე გზებიდან.

თვით ალგორითმის ჩამოყალიბებამდე ჯერ განვსაზღვროთ ძირითადი ცნებები. ქსელში დასაშვები სამარშრუტო ნაკადთა $x=\{x_p\}$ ვექტორისათვის $\Delta x=\{\Delta x_p\}$ ცვლილება უნდა აკმაყოფილებდეს შემდეგ პირობებს:

1. Δx უნდა იყოს დასაშვები მიმართულება. ეს ნიშნავს, რომ რ$\bar{\alpha}$მელიღაც $\bar{\alpha}$-სათვის ყველა $\alpha \in [0, \bar{\alpha}]$-სას უნდა აკმაყოფილებდეს შემდეგ პირობას:

$$\sum_{p \in P_w} \Delta x_p = 0, \quad w \in W,$$

$$x_p + \alpha \Delta x_p \geq 0, \quad \alpha \in [0, \alpha], \quad p \in P_w, \quad w \in W.$$

ეს პირობა აღნიშნავს იმ ფაქტს, რომ რომელიმე წყვილისათვის რომელიმე მარშრუტზე მონაცემთა ნაკადის გაზრდა კომპენსირებულ უნდა იქნას სხვა მარშრუტზე ნაკადის შემცირებით.

2. Δx უნდა იყოს დაღმასვლის მიმართულება. ე.ი. ღირებულებითი ფუნქცია უნდა იკლებდეს Δx მიმართულებით, ე.ი. $D(x+\alpha\Delta x) > D(x)$. სხვა სიტყვებით რომ ვთქვათ, Δx ცვლილების ვექტორისა და $\nabla D(x)$ ღირებულებითი ფუნქციის გრადიენტის სკალარული ნამრავლი უნდა იყოს უარყოფითი.

$$\sum_{w \in W} \sum_{p \in P_w} (\partial D(x)/\partial x_p) \Delta x_p < 0$$

მოცემული ალგორითმი იტერაციული ხასიათისაა, სადაც ყოველ იტერაციაზე $x=x+\alpha\Delta x$. α ყოველ იტერაციისათვის გამოითვლება ცალკე. დასაშვები დაღმასვლის მიმართულება x წერტილიდან არსებობს მხოლოდ იმ შემთხვევაში, თუ x არ არის ოპტიმალური.

ფრენკ-ვოლფის მეთოდი ემყარება დასაშვები დაღმასვლის მიმართულებების გასწვრივ მცირე ცვლილებების პრინციპს.

ვიპოვოთ უმცირესი პირველწარმოებული სიგრძის გზა $x=\{x_p\}$ მოცემული დასაშვები სამარშრუტო ნაკადების ვექტორისათვის კვანძების ყოველი წყვილისათვის, ხოლო $\bar{x} = \{\bar{x}_p\}$ არის სამარშრუტო ნაკადთა ვექტორი, რომელიც წარმოიქმნება თუ ყოველი $w \in W$ წყვილისათვის r_w შემომავალი ნაკადი გადაიცემა მინიმალური პირველწარმოებული სიგრძის მქონე გზით.

დავუშვათ, რომ α^* ბიჯის სიდიდეა ისეთი, რომ

$$D[x + a^*(\bar{x} - x)] = \min D[x + a^*(\bar{x} - x)]$$

$\alpha \in [0,1]$

მაშინ სამარშრუტო ნაკადთა მნიშვნელობები შემდეგ ბიჯზე იქნება

$$x_p = x_p - \alpha^*(\bar{x}_p - x_p),\ p \in P_w,\ w \in W$$

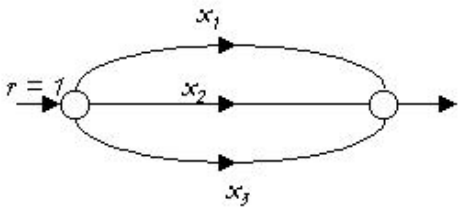
და ეს პროცესი კვლავ მეორდება. მოცემული მეთოდი ზღვარში ამცირებს ღირებულებითი ფუნქციის მნიშვნელობას მინიმუმამდე, მაგრამ ოპტიმალურ მნიშვნელობასთან მიახლოვებისას მისი კრებადობის სიჩქარე საკმაოდ მცირდება.

აღსანიშნავია, რომ ყოველი ნაკადის α^* წილი. რომელიც გადის არაუმოკლესი გზით ($x_p=0$), გადაიტანება უმოკლეს გზაზე ($x_p=r_w$) ყოველი $w \in W$ წყვილისათვის.

მოვიყვანოთ ფრენკ-ვოლფის მეთოდის გამოყენების ერთი მაგალითი. მოცემულია ქსელი, რომელიც შესდგება ორი კვანძისაგან: წყაროსაგან და მიმღებისაგან და მათი შემაერთებელი 3 ხაზისაგან, რომელთა ნაკადებია შესაბამისად x_1, x_2 და x_3 და რომლებიც აკმაყოფილებენ პირობებს:

$$x_1+x_2+x_3=1,\ x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0.$$

ხოლო ღირებულებით ფუნქციას აქვს შემდეგი სახე:



$$D(x)=(1/2)(x_1^2+x_2^2+0,1x_3^2)+0,55x_3.$$

ვიპოვოთ ამონახსნი ჯერ ანალიზური მეთოდით: მტკიცდება, რომ $x^*=\{x_1^*, x_2^*, x_3^*\}$ ოპტიმალური ამონახსნისათვის $x_1^*=x_2^*$, ამიტომ შესაძლოა შემდეგი ორი შემთხვევა: ა) $x_3^*=0$ და $x_1^*=x_2^*=1/2$; ბ) $x_3^*=\beta$ და $x_1^*=x_2^*=(1-\beta)/2$. ეს უკანასკნელი შემთხვევა შეუძლებელია, რადგან თუ $x_3^*>0$, ეს ნიშნავს, რომ $\partial D(x^*)/\partial x_3 \leq \partial D(x^*)/\partial x_1$, ანუ $0,1\beta+0,55 \leq (1-\beta)/2$, რაც აშკარად შეუძლებელია. ამიტომ ოპტიმალური ამონახსნია $x^*=(1/2,1/2,0)$.

ახლა განვიხილოთ ამ ამოცანის ამოხსნის პროცესი ფრენკ-ვოლფის მეთოდით. თითოეული ხაზის პირველწარმოებული სიგრძე იქნება:

$$\partial D(x)/\partial x_1=x_1,\ \partial D(x)/\partial x_2=x_2,\ \partial D(x)/\partial x_3=0,1x_3+0,55.$$

ცხადია, რომ უმოკლესი გზა იქნება ან x_1 ან x_2, ე.ი. იმისდა მიხედვით $x_1 \leq x_2$, თუ $x_1>x_2$, შესაბამისი ნაკადი უმოკლესი გზით იქნება $\bar{x}=(1,0,0)$ ან $\bar{x}=(0,1,0)$. ამიტომ ყოველ იტერაციას ექნება შემდეგი სახე:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} := \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \alpha^* \begin{bmatrix} 1-x_1 \\ -x_2 \\ -x_3 \end{bmatrix} = \begin{bmatrix} x_1+\alpha^*(x_2+x_3) \\ (1-\alpha^*)x_2 \\ (1-\alpha^*)x_3 \end{bmatrix},\ x_1 \leq x_2$$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} := \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \alpha^* \begin{bmatrix} -x_1 \\ 1-x_2 \\ -x_3 \end{bmatrix} = \begin{bmatrix} (1-\alpha^*)x_1 \\ x_2+\alpha^*(x_1+x_3) \\ (1-\alpha^*)x_3 \end{bmatrix},\ x_1 > x_2$$

ამგვარად ყოველ იტერაციაზე არაუმოკლესი გზებზე გამავალი ნაკადის α^* წილი გადაიტანება უმოკლეს გზაზე.

α^* გამოითვლება $D(x)$ ფუნქციის მინიმიზაციით [0,1] შუალედზე.

$D[x+(\bar{x}-x)]=(1/2)([x_1+\alpha(\bar{x_1}-x_1)]^2+[x_2+\alpha(\bar{x_2}-x_2)]^2+$

$+0,1[x_3+\alpha(\bar{x}_3-x_3)]^2+0,55[x_3+\alpha(\bar{x}_3-x_3)]).$

კერძოდ თუ ფუნქციას გავაწარმოებთ α-ს მიმართ და გავუტოლებთ 0-ს, ვიპოვით $\bar{\alpha}$ α-ის მნიშვნელობას, რომლის დროსაც მიიღწევა ფუნქციის $D[x+\alpha(\bar{x}-x)]$ მინიმუმი:

$$\bar{\alpha} = - \frac{x_1(\bar{x}_1-x_1)+x_2(\bar{x}_2-x_2)+(0,1x_3+0,55)(\bar{x}_3-x_3)}{(\bar{x}_1-x_1)^2+(\bar{x}_2-x_2)^2+0,1(\bar{x}_3-x_3)^2}$$

$\bar{\alpha} \geq 0$, ამიტომ თუ გავითვალისწინებთ, რომ $\alpha \in [0,1]$, მივიღებთ $\alpha^* = \min[1, \bar{\alpha}]$. შემდეგ ტაბულაში მოცემულია იტერაციების სურათი ფრენკ-ვოლფის მეთოდის გამოყენებისას:

იტერაცია k	0	10	20	40	80	160	320
x_1	0,4	0,4593	0,4702	0,4795	0,4866	0,4917	0,4950
x_2	0,3	0,4345	0,4562	0,4717	0,4823	0,4893	0,4938
x_3	0,3	0,1061	0,0735	0,0790	0,0310	0,0189	0,0110
ღირებულება	0,2945	0,2588	0,2553	0,2532	0,2518	0,2510	0,2506
ცდომილებათა შეფარდება $\frac{D(x^{k+1})-D(x^*)}{D(x^k)-D(x^*)}$	0,7164	0,9231	0,9576	0,9774	0,9882	0,9939	0,9969

როგოც ამ ტაბულიდან ჩანს, კრებადობის სიჩქარე მნიშვნელოვნად კლებულობს ოპტიმუმის სიახლოვეში.

3. მარშრუტიზაციის ალგორითმების შედარებითი ანალიზი

წინა თავში იყო მოყვანილი მარშრუტიზაციაში მონაწილე სხვადასხვა ალგორითმების ფორმალური აღწერები. ახლა დავახასიათოთ ისინი და შევადაროთ ერთმანეთს, განვიხილოთ მათი დადებითი და უარყოფითი მხარეები.

როგორც ბელმან-ფორდის ალგორითმის აღწერიდან ჩანს, მის მუშაობისას ყოველი კვანძისათვის შეიძლება ადგილი ჰქონდეს ყველაზე უფრო დიდი $N-1$ იტერაციას. ეს იტერაციები სრულდება $N-1$ კვანძისათვის, ხოლო ყოველი კვანზისათვის მინიმიზაცია ხდება ყველაზე მეტი $N-1$ ცვლადის მიხედვით. ამიტომ ამ ალგორითმისათვის გამოთვლების მოცულობა იზრდება კვანძთა რაოდენობის კუბის პროპორციულად:

$$V \sim F(N^3).$$

მაგრამ რეალურად, პრაქტიკაში საჭიროა გამოთვლების გაცილებით ნაკლები სიმძლავრე, რადგან თითოეული კვანძისათვის ალგორითმი ასრულებს $N-1$-ზე ნაკლებ იტერაციას. გარდა ამისა, რკალების რაოდენობა გაცილებით ნაკლებია N^2-ზე. ეს იწვევს ბელმან-ფორდის ალგორითმის პრაქრიკაში გამოყენებისას გამოთვლების მნიშვნელოვან შემცირებას.

გარდა ამისა მეტად მნიშვნელოვანია შეზღუდვა, რომ ქსელი არ უნდა შეიცავდეს უარყოფითი სიგრძის ციკლებს, რადგან ამ შემთხვევაში აღნიშნული ციკლის ყოველ გავლასთან ერთად მარშრუტის საერთო სიგრძე მცირდება, ამიტომ ალგორითმი არასოდეს არ დაამთავრებს თავის მუშაობას. უარყოფითი სიგრძის ციკლების აღმოჩენა შეიძლება მოხდეს შემდეგი მეთოდით: ყოველი i კვანძისათვის შედარდეს $D_i^{(n)}$ და $D_i^{(n-1)}$ სიდიდეები. თუ ისინი ყველა i-სათვის ტოლია, მაშინ უარყოფითი სიგრძის ციკლებს ადგილი არ აქვთ.

დიკსტრას ალგორითმის შემთხვევაში თითოეულ ბიჯზე გამოთვლების რაოდენობა პროპორციულია N-ის, ხოლო იტერაციათა რაოდენობა არის $N-1$. ამიტომ გამოთვლების მოცულობა უარეს შემთხვევაში პროპორციულია N^2-ის:

$$V \sim F(N^2).$$

მიუხედავად იმისა, რომ თეორიულად დიკსტრას ალგორითმი მოითხოვს გაცილებით ნაკლებ გამოთვლების მოცულობას, ხშირია შემთხვევები (როდესაც ქსელის სტრუქტურა შედარებით მარტივია), როცა ბელმან-ფორდის ალგორითმი უფრო მისაღებია. ეს განსაკუთრებით იგრძნობა კვანძების შემაერთებელი ხაზების ნაკლებობის შემთხვევაში.

ფლოიდ-უორშელის ალგორითმში უმოკლესი გზა იძებნება ყოველი წყვილისათვის ცალკ-ცალკე, ხოლო ამ წყვილთა რაოდენობა N^2-ის პროპორციულია. ხოლო ყოველი წყვილისათვის ბიჯთა რაოდენობა N-ის ტოლია, ამიტომაც ჯამური გამოთვლების რაოდენობა ბელმან-ფორდის ალგორითმის მსგავსად N^3-ის პროპორციულია:

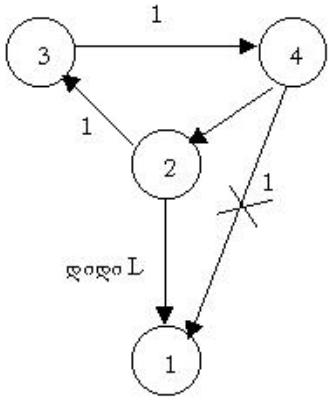
$$V \sim F(N^3).$$

ამგვარად ფლოიდ-უორშელის ალგორითმი მოითხოვს გამოთვლების ყველაზე დიდ მოცულოვას, რადგან ბელმან-ფორდის ალგორითმის მსგავსად მას არ გააჩნია შემთხვევები, როცა ის დროზე ადრე ამთავრებს თავის მუშაობას.

ბელმან-ფორდის განაწილებული ასინქრონული ალგორითმი წარმოადგენს ერთ-ერთ ყველაზე ეფექტურ ალგორითმს. ამ ალგორითმის თავისებურებას ის წარმოადგენს, რომ იგი შეიძლება მუშაობდეს პარალელურად სხვადასხვა კვანძში, რაც მნიშვნელოვნად ამცირებს გამოთვლების მოცულობას. გარდა ამისა, მოცემული ალგორითმის ფუნქციონირებისათვის ყოველ კვანძში საჭიროა გაცილებით ნაკლები ინფორმაციის შენახვა. თითოეულმა კვანძმა უნდა მხოლოდ

იცოდეს მისგან უშუალოდ გამავალი ხაზების სიგრძეები და ქსელში არსებული კვანძების ნომრები. მაგრამ ამ ალგორითმსაც გააჩნია უარყოფითი მხარეები.

განვიხილოთ შემთხვევა, როცა ქსელის სტრუქტურას აქვს შემდეგი სახე:



უმოკლესი გზებს დასაწყისში ექნებათ შემდეგი სახე: $\{D_2=3,\ D_3=2,\ D_4=1\}$, მას შემდეგ რაც ხაზი (4,1) გამოვა მწყობრიდან ალგორითმს დასჭირდება დაახლოვებით L იტერაცია, სანამ კვანძი 2 გაიგებს რომ 1 კვანძამდე უმოკლესი გზა არის ხაზი (2,1). სანამ კი ეს მოხდება, კვანძები 3 და 2 განაგრძობენ 1 კვანძისათვის განკუთვნილი პაკეტების გაგზავნას 4 კვანძის გავლით, ხოლო კვანძი 4 თვლის რომ უმოკლესი გზა 1 კვანძამდე ძევს 2 ან 3 კვანზების გავლით, და ამიტომ უბრუნებს მათ პაკეტებს. წარმოიქმნება ე.წ. სამარშრუტო მარყუჟი. (2,4) და (3,4) ხაზების სიგრძეები თანდათანობით იზრდებიან და სწორი მარშრუტიზაცია მიღწეული არ იქნება, სანამ მათი ფასები არ გაუტოლდება (2,1) ხაზის ფასს. ამ მოვლენას ეწოდება უსასრულობამდე თვლა (Count to Infinity). მისი თავიდან აცილების გზები განიხილება პროტკოლების აღწერებში.

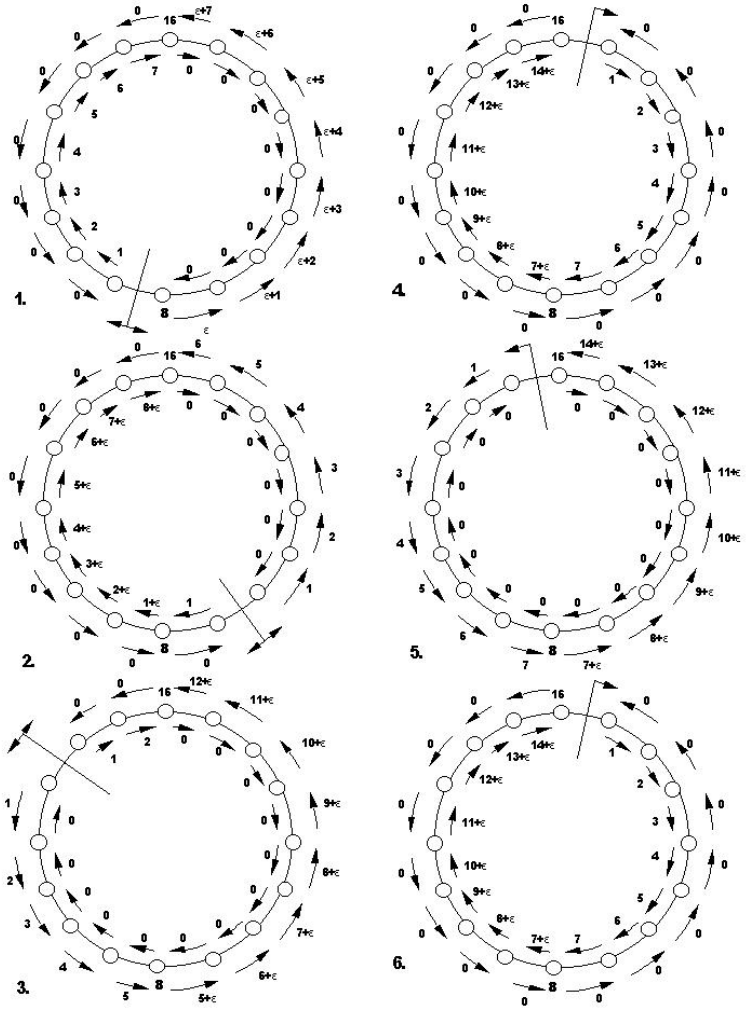
ცხადია, მეტად მოსახერხებელია, თუ თვითოეული ხაზის სიგრძეს დავუკავშირებთ მის ისეთ ფიზიკურ მახასიათებლებს, როგორიცაა ნაკადის სიმკვრივე. ამ შემთხვევაში ჩნდება მეტად მნიშვნელოვანი პრობლემა. ესაა სამარშრუტო რხევების პრობლემა. ეს იმით არის გამოწვეული, რომ თუ კვანძი მიმართავს ნაკადს უმოკლესი მარშრუტით, ამის შედეგად ამ მარშრუტის სიგრძე გაიზრდება და შესაძლებელია, რომ ის უკვე აღარ იქნება უმოკლესი, რაც გამოიწვევს მონაცემთა ნაკადის კვლავ ახალი მარშრუტით მიმართვას. რხევების მაღალი სიხშირისას კი მნიშვნელოვნად იზრდება პაკეტის დაყოვნების სიდიდე, რაც უარყოფითად მოქმედებს ქსელის ფუნქციონირებაზე.

მარშრუტიზაციის ამგვარი ქცევა განსაკუთრებითაა დამახასიათებელი დეიტაგრამული ქსელებისათვის, რადგან აქ ყოველი დეიტაგრამის მარშრუტიზაცია დამოუკიდებლად ხდება. ეს იწვევს იმ ფაქტს, რომ მარშრუტის შეცვლა უმალვე იწვევს მთელი მონაცემთა ნაკადის გადატანას ახალ მარშრუტზე.

ამ ნაკლის თავიდან აცილება შეიძლება, თუ ყოველი ხაზის სიგრძეს დავუმატებთ რაიმე α მუდმივ რიცხვს, ისე რომ $d_{ij}=\alpha > 0$, როცა $F_{ij}=0$. რაც უფრო დიდია α, მით უფრო ნაკლებია სამარშრუტო რხევები ქსელში, მაგრამ სამაგიეროდ მით უფრო ნაკლებად მგრძნობელი ხდება ქსელი ხაზების გადატვირთვის მიმართ.

დეიტაგრამულ ქსელებში სამარშრუტო რხევების საილუსტრაციოდ განვიხილოთ ერთი მაგალითი. დავუშვათ მოცემული 16 კვანძიანი ქსელი, რომელსაც გააჩნია წრიული ტოპოლოგია. ამ ქსელში მხოლოდ ერთი მიმდებია (კვანძი 16). ყოველ დანარჩენ კვანძში შემავალი ნაკადის სიდიდე 1-ის ტოლია, მხოლოდ მე-8 კვანძში შედის $\varepsilon > 0$ ნაკადი, სადაც ε მეტად მცირეა. თითოეული

ხაზის სიგრძე ტოლია ხაზში შემომავალი ნაკადის $d_{ij} = F_{ij}$, ე.ი. კვანძში შემომავალი და ტრანზიტული ნაკადების ჯამი. დავუშვათ, რომ ყოველი კვანძი T დროის განმავლოვაში ითვლის F_{ij} ნაკადის იტენსიურობებს, ხოლო შემდეგი T დროის განმავლობაში ხმარობს მიღებულ შედეგებს. ვთქვათ დასაწყისში კვანძები 1-7 ირჩევენ მარშრუტს საათის ისრის მიმართულებით, ხოლო 8-15 - საწინააღმდეგო მიმართულებით, რაც წარმოადგენს საკმაოდ კარგ მარშრუტიზაციას. როგორც სქემიდან ჩანს, მარშრუტების სამი განახლების შემდეგ მიიღწევა მონაცემთა ნაკადის მუდმივი რხევები, როცა მთელი ტრაფიკი გადაიცემა ხან საათის ისრის და ხან საწინააღმდეგო მიმართულებით.

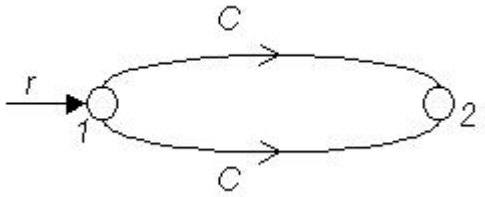


განვიხილოთ პირველი იტერაცია. აქ 8 კვანძისათვის შესაბამისი ორი ალტერნატიული გზის სიგრძეებია *28* და *28+8ε*. ამიტომ ის ირჩევს საათის ისრის მიმართულებას. 7 კვანძისათვის ეს სიგრძეები იქნება *28* და *28+7ε*, ამიტომ ის აგრეთვე აირჩევს საათის ისრის მიმართულებას. სხვა დანარჩენი კვანძისათვის უმოკლესი მარშრუტები არ შეიცვლება და ამიტომაც მონაცემთა გადაცემის მიმართულებაც იგივე დარჩება.

ამგვარი რხევები ახასიათებს ვირტუალურ წრედებიან ქსლებსაც, მაგრამ შედარებით ნაკლებად. აქ მონაცემთა გადაცემის ყოველ სეანსს მიეწერება გარკვაული მარშრუტი, რომელიც უცვლელი რჩება სეანსის დამთავრებამდე. ამის გამო, ქსელის რეაქცია უმოკლესი მარშრუტების შეცვლაზე არ არის ასეთი მკვეთრი, რადგან ძველი სეანსები განაგრძობენ ძველი უმოკლესი გზების გამოყენებას.

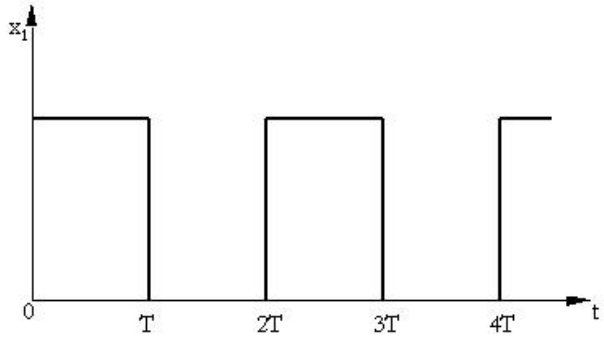
ქსელში მარშრუტიზაციის ქცევის საილუსტრაციოდ განვიხილოთ კიდევ ერთი მაგალითი. მოცემულია მარტივი ქსელი, რომელიც შესდგება წყაროსა და მიმღებისაგან, რომლებიც შეერთებულნი არიან ერთნაირი C გამტარუნარიანობის მქონე ორი ხაზით. წყაროში შემოდის მონაცემთა r ნაკადი.

ცხადია, ოპტიმალური მარშრუტიზაციისას r ნაკადი უნდა გაიყოს ორ ტოლ ნაწილად, ამიტომ ქსელის ჯამური გამტარუნარიანობა შეიძლება აღწევდეს $2C$-ს.



მარშრუტიზაციის ალგორითმი ზომავს ორივე ხაზისათვის საშუალო დაყოვნებებს T დროის განმავლობაში და შემდეგი T დროის ინტერვალის განმავლობაში გადააგზავნის ტრაფიკს წინა ინტერვალში ნაკლებად დატვირთული ხაზით.

დეიტაგრამული ქსელის შემთხვევაში ერთ-ერთ ხაზში დროის ინტერვალის განმავლობაში ხან გაივლის მთელი შემომავალი ნაკადი, ხან კი არაფერი არ გაივლის.

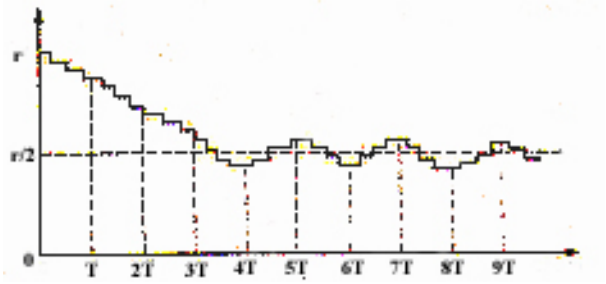


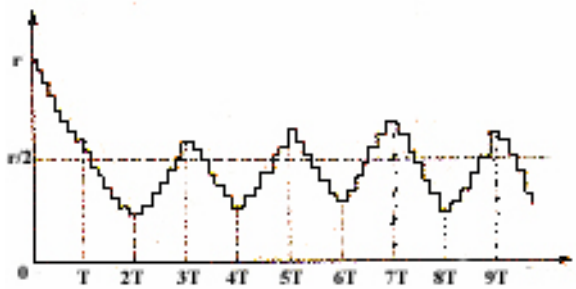
განვიხილოთ ვირტუალურ წრედებიანი ქსელის შემთხვევა. ვთქვათ ვირტუალური წრედების შექმნის მომენტები ექვემდებარებიან პუასონის განაწილებას λ სიმკვრივით და ყოველი წრედი თავისი არსებობის დროის განმავლობაში იყენებს გზას, რომელიც მას დაუნიშნა მარშრუტიზაციის ალგორითმმა. დავუშვათ ვირტუალური ქსელის არსებობის საშუალო ხანგრძლივობაა $1/\mu$ წმ და განაწილებულია ექსპონენციალური კანონით. აქედან გამომდინარე მოცემულ მომენტში არსებული ვირტუალური ქსელების რაოდენობა განაწილდება პუასონის კანონით და მათი საშუალო რაოდენობა იქნება λ/μ. თუ γ არის ვირტუალურ ქსელში მონაცემთა გადაცემის საშუალო სიჩქარე, მაშინ $r=(\lambda/\mu)\gamma$ ანუ

$$\gamma = r\mu / \lambda$$

დავუშვათ, რომ უმოკლესი გზების განახლების T პერიოდი ვირტუალური წრედის $1/\mu$ არსებობის საშუალო დროსთან შედარებით მცირეა. მაშინ T წამიანი ინტევალის განმავლობაში მოსპობილი ვირტუალური წრედების წილი იქნება μT, ხოლო ახლად შექმნილი ვირტუალური წრედების რაოდენობა იქნება λT. ახალი ვირტუალური წრედები გაზრდიან ხაზებში ნაკადთა ინტენსივობას $r\mu T$ სიდიდით. ამის გათვალისწინებით, თითოეულ ხაზში k-ურ დროით ინტერვალში საშუალო ინტენსივობათა სიდიდეები შეიცვლება შემდეგი კანონით:

$$x_i^{k+1} = \begin{cases} (1-\mu T)x_i^k + r\mu T, & \text{თუ } x_i^k = \min\{x_1^k, x_2^k\} \\ (1-\mu T)x_i^k & \text{წინააღმდეგ შემთხვევაში} \end{cases}$$





თუ განვიხილავთ ქსელის ხაზებში დატვირთვების გრაფიკებს, დავინახავთ, რომ ხაზების დატვირთვის ინტენსივობა მერყეობს $r/2$ სიდიდის გარშემო დაახლოვებით $r\mu T$ ამპლიტუდით. აქედან გამომდინარე ალგორითმი იძლევა საუკეთესო შედეგებს, როცა ვირტუალური წრედების არსებობის საშუალო დრო გაცილებით მეტია უმოკლესი გზების განახლების პერიოდზე ($\mu T << 1$) და იდეალურ შემთხვევაში თითოეულ ხაზში დატვირთვის ინტენსივობა $r/2$-ის ტოლია.

ამგვარად როგორც დავინახეთ, დინამიური მარშრუტიზაციისას ერთ-ერთი ყველაზე მნიშვნელოვანი პარამეტრია ვირტუალური წრედის არსებობის საშუალო დრო. თუ ეს სიდიდე მცირეა, მაშინ უმოკლესი გზების განახლების სიხშირეც უნდა გაიზარდოს, მაგრამ ამ უკანასკნელის გაზრდა შეიძლება გარკვეულ ზღვრამდე, რადგან ხაზების სიგრძეების შეცვლა საჭიროებს რაღაც სასრულ დროს.

სამარშრუტო ინფორმაციის ალგორითმებს შორის ქსელებში ძირითადად გამოიყენება ზვავური ალგორითმი პერიოდული განახლებებით. ამ ალგორითმის მთავარ ნაკლს წარმოადგენს რიგითი ნომრის ველის გადავსების შესაძლებლობა. მიუხედავად იმისა, რომ რიგითი ნომრის ველის სიდიდის გაზრდა მნიშვნელოვნად

ამცირებს ამ მოვლენის მოხდენის ალბათობას, მაგრამ ყოველთვის შესაძლებელი რჩება ინფორმაციის გადაცემისას და კვანძის მეხსიერებაში მისი შენახვისას რიგითი ნომრების დამახინჯება. ამ პრობლემის ზემოთ მოყვანილი მეთოდით გადაჭრისას მიმდინარე რიგითი ნომერი შეიძლება მკვეთრად გაიზარდოს, რამაც შესაძლოა გამოიწვიოს რიგითი ნომრის ველის გადავსება.

მართალია სამარშრუტო ინფორმაციის გავრცელებისას შეტყობინებების გაგზავნის პერიოდულობა ზრდის ქსელში მონაცემთა ნაკადთა ინტენსივობებს, მაგრამ სამაგიეროდ ეს მეთოდი გამოირჩევა გაცილებით უფრო მეტი საიმედობით, რადგან შესაძლებელია შეტყობინებათა კარგვა მათი ქსელში გადაცემის პროცესში.

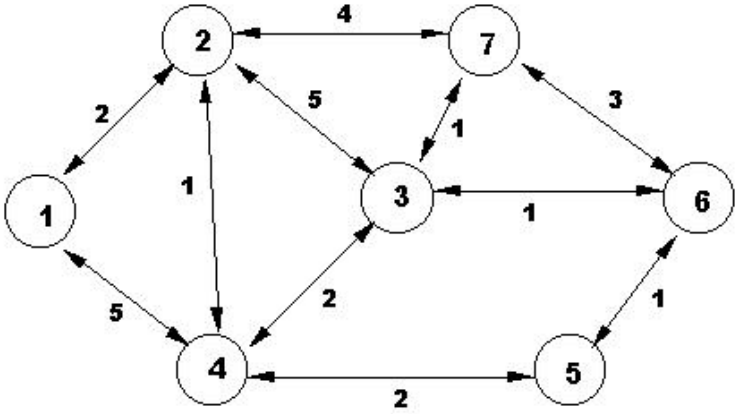
ალგორითმი რიგითი ნომრების გამოყენების გარეშე გამოიყურება მეტად მომხიბვლელად, რადგან ხსნის რიგითი ნომრებისა და ასაკის ველების გადავსებასთან და ქსლის მიერ ბმულობის დაკარგვისას რიგითი ნომრების "დავიწყების" პრობლემებს. გარდა ამისა, ალგორითმი რიგითი ნომრების გარეშე რეაგირებს კონკრეტულ მოვლენებზე და არ აწარმოებს ინფორმაციის პერიოდულ გაგზავნას, რაც ამცირებს ქსელის ხაზების დატვირთვას. მაგრამ ყოველივე ამას დადებით მხარეებთან ერთად გააჩნია უარყოფითი მხარეებიც. ალგორითმი ნაკლებად მედეგიანია ინფორმაციის გადაცემისას და შენახვისას მისი დამახინჯებისა და დაკარგვის მიმართ. ეს ნაკლი გააჩნია ყველა ალგორითმს, რომელიც არ იყენებს შეტყობინებების პერიოდულ გაგზავნას. გარდა ამისა, ეს ალგორითმი უფრო რთულია ზვავურ ალგორითმთან შედარებით და მოითხოვს მეტ პროცესორულ სიმძლავრეებსა და მეხსიერების მოცულობას, რადგან საჭიროა არა მარტო ძირითადი ტოპოლოგიური ტაბულის შენახვა, არამედ საპორტო ტოპოლოგიური ტაბულების განთავსებაც ყოველი მეზობლისათვის, რაც მეზობლების დიდი რაოდენობის შემთხვევაში მეხსიერების დიდ ხარჯებთან არის დაკავშირებული.

ოპტიმალური მარშრუტიზაციის ალგორითმები თავინთი მახასიათებლებით გაცილებით ჯობია უმოკლესი გზების ალგორითმებს. მათ სრულებით არ ახასიათებთ სამარშრუტო რხევები და ისინი იძლევიან საშუალებას შემომავალი მონაცემთა ნაკადი ეფექტურად განაწილდეს რამოდენიმე ალტერნატიულ გზას შორის. გარდა ამისა, ღირებულებითი ფუნქციის ცვლილებით შესაძლებელი ხდება მოცემული ალგორითმის ადაპტირება კონკრეტული მოთხოვნებისათვის.

ფრენკ-ვოლფის მეთოდის გამოთვლების მოცულობის ცალსახა შეფასება შეუძლებელია, რადგან აქ იტერაციათა რაოდენობა არ არის შეზღუდული და ყოველი კონკრეტული ამოცანიდან გამომდინარე უნდა შერჩეულ იქნას კომპრომისი იტერაციათა რიცხვსა და ალგორითმის ოპტიმუმთან სიახლოვეს შორის. მაგრამ ყველა შემთხვევაში მოცემული ალგორითმი გაცილებით უფრო რთულია, ვიდრე უმოკლესი გზის პოვნის ალგორითმები.

4. მოცემული ალგორითმების პროგრამული რეალიზაციის მაგალითები

მოვიყვანოთ ზემოთ განხილული ალგორითმების პროგრამული რეალიზაციები. ისინი შესრულებულნი არიან დაპროგრამების ენაზე C, და ძირითადად გააჩნიათ საილუსტრაციო ხასიათი. ქვემოთ მოყვანილია ბელმან-ფორდის და დიკსტრას ალგორითმების შესაბამისი პროგრამული კოდი. ეს პროგრამები დაწერილია სქემაზე მოყვანილი იდეალური ქსელისათვის უმოკლესი მარშრუტების გამოსათვლელად.



ეს პროგრამები შექმნილია იმ დაშვებით, რომ ქსელის კვანძმა, რომელშიც მუშაობს ეს ალგორითმი, უკვე იცის მთელი ქსელის ტოპოლოგია და ქსელის ტოპოლოგია უცვლელი დარჩება ალგორითმის მოქმედების დროის შუალედში. მოყვენილი პროგრამული რეალიზაციები შემუშავებულია ერთ კვანძში შესასრულებლად და არ ითვალისწინებენ განაწილებული ალგორითმების თავისებურებებს, როგორიცაა გამოთვლების შედეგების მიმოცვლა და სინქრონიზაცია.

```
#include <stdio.h>
#include <iostream.h>

const short INF = 255;
const short NODE_NUM = 7;

short net [NODE_NUM][NODE_NUM]={{0,2,INF,5,INF,INF,INF},\
                                {2,0,5,1,INF,INF,4},\
                {INF,5,0,3,INF,1,INF},\
                {5,1,3,0,2,INF,INF},\
                {INF,INF,INF,2,0,1,INF},\
                {INF,INF,1,INF,1,0,3},\
                {INF,4,INF,INF,INF,3,0}};

short Bellman_Ford (short, short);

short Bellman_Ford (short src, short dst)
   {
     short metric = INF;
```

```
     if (src==dst) return 0;
     for (int w=0; w<NODE_NUM; w++)
       {
         if ((w==src)||(w==dst)) continue;
         if (net[dst][w]==INF) continue;
         short templen = Bellman_Ford (src,w);
         if (net[dst][w]+templen < metric)
            metric = net[dst][w]+templen;
       }
      return metric;
   }


int main ()
   {
     cout<<"Network Metric Table"<<endl<<endl;
     for (int i=0; i<NODE_NUM;i++)
        {
          for (int j=0; j<NODE_NUM;j++)
            {
             short SP;
             SP = Bellman_Ford(i,j);
             cout<<SP<<"  ";
            }
          cout<<endl;
        }
      return 0;
   }

#include <stdio.h>
#include <iostream.h>

const short INF = 255;
const short NODE_NUM = 7;

                short net [NODE_NUM][NODE_NUM]=
                {{0,2,INF,5,INF,INF,INF},\
                {2,0,5,1,INF,INF,4},\
                {INF,5,0,3,INF,1,INF},\
                {5,1,3,0,2,INF,INF},\
                {INF,INF,INF,2,0,1,INF},\
                {INF,INF,1,INF,1,0,3},\
                {INF,4,INF,INF,INF,3,0}};
short Dijkstra (short, short);

short Dijkstra (short src, short dst)
   {
    short minnode, minlen = INF;
    short metric [NODE_NUM];
```

```
    bool cont = true;
    bool considered [NODE_NUM];
    if (src == dst) return 0;
    for (int i=0; i<NODE_NUM;i++)
       {
        metric[i] = net[src][i];
        if (i==src) considered[i] = true;
          else considered [i] = false;
       }
    while (cont){
      cont = false;
      for (short i = 0;i<NODE_NUM;i++)
        {
         if (!considered[i]){
           cont = true;
           short temp = Dijkstra(src,i);
           if (temp<minlen){
             minnode = i;
             minlen = temp;
             }
           considered[minnode] = true;
         }
        }
        for (int j = 0, j<NODE_NUM,j++)
          {
           if (!considered[j]&&(minlen+net[i][j]<metric[j])){
            metric[j] = minlen + net[i][j];
           }
          }
    }
    return metric[dst];
   }

   int main ()
   {
     cout<<"Network Metric Table"<<endl<<endl;
     for (int i=0; i<NODE_NUM;i++)
        {
          for (int j=0; j<NODE_NUM;j++)
            {
             short SP;
             SP = Dijkstra(i,j);
             cout<<SP<<"  ";
            }
          cout<<endl;
        }
      return 0;
   }
```

ორივე პროგრამისათვის მიღებულია შედეგი, რომელიც წარმოადგენს ზემოთ მოყვანილი ქსელის უმოკლესი გზების ტაბულას:

Network Metric Table

0	2	5	3	5	6	6
2	0	3	1	3	4	4
5	3	0	2	2	1	1
3	1	2	0	2	3	3
5	3	2	2	0	1	3
6	4	1	1	3	0	2
6	4	1	3	3	2	0

5. დინამიური მარშრუტიზაციის პროტოკოლები

როდესაც ქსელი აღწევს გარკვეულ სირთულეს, მასში ვეღარ ხერხდება ეფექტური მარშრუტიზაციის განხორციელება სტატიკური მარშრუტიზაციის მეთოდებით. სწორედ ამ შემთხვევაში ხდება საჭირო დინამიური მარშრუტიზაცია. ქსელებში დინამიური მარშრუტიზაციის ალგორითმების რეალიზაცია ხდება დინამიური მარშრუტიზაციის პროტოკოლების მეშვეობით.

არსებობს მრავალი დინამიური მარშრუტიზაციის პროტოკოლი, რომლებიც ემყარებიან სხვადასხვა მოქმედების პრინციპებს და კრიტერიუმებს. ინტერნეტის მთელი მარშრუტიზაციის სისტემა ეყრდნობა იმ ფაქტს, რომ ინტერნეტი დაყოფილია ე.წ. ავტონომიურ სისტემებად. ავტონომიური სისტემა - ეს არის ქსელების ისეთი ერთობლიობა, რომლებიც ექვემდებარებიან ერთიან ადმინისტრირებას. ამიტომ განასხვავებენ დინამიური მარშრუტიზაციის პროტოკოლების ორ კლასს: შიდა მარშრუტიზაციის პროტოკოლები (Interior Gateway Protocol) და გარე მარშრუტიზაციის პროტოკოლები (Exterior Gateway Protocol). შიდა მარშრუტიზაციის პროტოკოლები უზრუნველყოფენ მონაცემთა პაკეტების გადაადგილებას ავტონომიური სისტემის შიგნით, ხოლო გარე მარშრუტიზაციის პროტოკოლები ემსახურებიან ავტონომიური სისტემის ზოგი შიდა მარშრუტის გაცემას მის ფარგლებს გარეთ, ანუ მარშრუტების მიმოცვლას სხვადასხვა ავტონომიურ სისტემებს შორის.

ახლა განვიხილოთ რამოდენიმე ყველაზე მნიშვნელოვანი როგორც გარე, ისე შიდა მარშრუტიზაციის პროტოკოლი და მოვახდინოთ მათი ძირითადი მახასიათებლების შედარება.

5.1. RIP

RIP (Routing Information Protocol) პროტოკოლი შეიქმნა 1982 წელს და წარმოადგენს TCP/IP ქსელებში დინამიური მარშრუტიზაციის საფუძველს. თავისი პოპულარობა მან მოიპოვა იმის გამო, რომ RIP-ის ერთ-ერთი რეალიზაცია შეტანილ იქნა BSD UNIX-ის დისტრიბუტივში. ის დეტალურადაა აღწერილი RFC 1058 დოკუმენტში.

ეს პროტოკოლი მიეკუთვნება დისტანციურ-ვექტორული პროტოკოლების კლასს და წარმოადგენს შიდა მარშრუტიზაციის პროტოკოლების ერთ-ერთ კლასიკურ რეალიზაციას. პროტოკოლი მარშრუტის სიგრძის კრიტერიუმად ხმარობს მიმღებამდე ნახტომების (hops) რაოდენობას, ე.ი. თუ რამდენი საშუალედო მარშრუტიზატორი უნდა გაიაროს საინფორმაციო პაკეტმა, სანამ მიაღწევს მიმღებს. მარშრუტის სიგრძე შეიძლება იღებდეს მნიშვნელობებს 1-15 შუალედიდან. RIP პროტოკოლი ემყარება უმოკლესი გზების გამოთვლას ბელმან-ფორდის (სხვანაირად ფორდ-ფულკერსონის) განაწილებულ ალგორითმს, ხოლო განახლების შეტყობინებების გავრცელებისას ხმარობს ზვავურ ალგორითმს პერიოდული გადაგზავნებით. მისი სამარშრუტო ტაბულის ყოველ ჩანაწერში ინახება შემდეგი ინფორმაცია:

მიმღები	შემდეგი მარსრუტიზატორი	მარშრუტის სიგრძე	ტაიმერები	ალმები

გარდა ამისა ყოველ კვანძში ადმინისტრატორი შეიძლება უთითებდეს ქვექსელების ნიღბებს (რეალიზებული მხოლოდ RIP 2 რეალიზაციაში), რაც იძლევა შესაძლებლობას გამოყენებულ იქნას CIDR ტიპის დამისამართება. RIP ინახავს მხოლოდ საუკეთესო მარშრუტებს. თუ ახალი ინფორმაცია იძლევა უკეთეს მარშრუტს, ეს უკანასკნელი ჩაანაცვლებს უკვე არსებულს. ამის შემდეგ პროტოკოლი უგზავნის განახლების შეტყობინებებს თავის სხვა მეზობლებს. განახლების შეტყობინებას აქვს შემდეგი ფორმატი:

პირველი ბრძანების ველი შეიძლება იღებდეს შემდეგ მნიშვნელობებს:

1 - *მოთხოვნა* მოპასუხე სისტემისაგან გამოაგზავნოს მთლიანი სამარშრუტო ტაბულა ან მისი ნაწილი;

2 - *პასუხი*, შეიცავს გამგზავნის სამარშრუტო ტაბულას ან მის ნაწილს. ის შეიძლება გაგზავნილ იქნას მოთხოვნის საპასუხოდ, ან შეიძლება წარმოადგენდეს გამგზავნის მიერ გენერირებულ განახლების შეტყობინებას.

ამ ველის დანარჩენი მნიშვნელობები ძირითადად იგნორირდება და გააჩნია რაიმე კონკრეტული მნიშვნელობა მხოლოდ ცალკეული მწარმოებლების რეალიზაციებში.

ვერსიის ველი აღნიშნავს პროტოკოლის რეალიზაციის ტიპს და გამოიყენება არათავსებადი რეალიზაციების აღმოსაჩენად. ამ მომენტისათვის ცნობილია RIP 1 და RIP2.

პროტოკოლი შეიძლება აწარმოებდეს მარშრუტიზაციას სხვადასხვა ტიპის ქსელებში. ქსელის ტიპის საიდენტიფიკაციოდ არის განკუთვნილი მისამართის ოჯახის იდენტიფიკატორი. ჯერ-ჯერობით შექმნილია რეალიზაცია მხოლოდ TCP/IP ქსელებისათვის, რომლისთვისაც ველის მნიშვნელობა 2-ის ტოლია.

შემდეგია IP მისამართის 4 ბაიტიანი ველი. ეს შეიძლება იყოს როგორც დანიშნულების ქსელის, ისე ცალკეული სადგურის მისამართი.

ბოლოს მიეთითება მარშრუტის სიგრძის ველი, რომელიც შეიძლება იღებდეს მნიშვნელობებს 1-16 შუალედიდან და ახასიათებს დანიშნულების კვანძამდე საშუალედო კვანძების რაოდენობას.

შეტყობინების მონაკვეთი მისამართის ოჯახის იდენტიფიკატორიდან მარშრუტის სიგრძემდე შეიძლება რამოდენიმეჯერ გამეორდეს შეტყობინებაში, თითოჯერ ყოველი დანიშნულების ადგილისათვის. მაგრამ ამ გამეორებათა რაოდენობა არ შეიძლება აღემატებოდეს 25-ს.

RIP პროტოკოლი თავისი შეტყობინებების გადაცემისათვის იყენებს UDP პროტოკოლს. ყველა შეტყობინება იგზავნება 520 პორტზე. ვინაიდან დეიტაგრამის სიდიდე ვერ აღემატება 512 ბაიტს, ამიტომ ამაზე დიდი შეტყობინების გადასაცემად ფორმირდება რამოდენიმე RIP შეტყობინება.

RIP პროტოკოლის სამარშრუტო ტაბულის ყოველ ჩანაწერთან დაკავშირებულია რამოდენიმე ტაიმერი:

პერიოდულ განახლებათა ტაიმერი (routing update timer) განსაზღვრავს დროის მომენტებს როდესაც კვანძი გადასცემს საკუთარი სამარშრუტო ტაბულების ასლებს თავის მეზობლებს. მისი პერიოდია 30 წმ. ხშირად ხდება ხოლმე, რომ მთელი რიგ კვანძების განახლებათა ტაიმერები სინქრონიზირდება. ამ შემთხვევაში დროის გარკვეულ მომენტში ყველა ეს კვანძი იწყებს თავიანთი სამარშრუტო ტაბულების გადაცემას, რაც იწვევს ქსელის დატვირთვის მნიშვნელოვან გაზრდას. ამ პასუხის შეტყობინების მიღებისას პროტოკოლი ამოწმებს პირველ რიგში ვერსიის ველს. თუ ის 0-ის ტოლია, მაშინ ხდება ამ შეტყობინების იგნორირება. თუ ის 1-ია, მაშინ მოწმდება ნულოვანი ველები. თუ ისინი შეიცავენ ნულისაგან განსხვავებულ მნიშვნელობებს, შეტყობინება იგნორირდება. თუ ვერსიის ველი მეტია 1-ზე, მაშინ ნულოვანი ველების შიგთავსი იგნოროდება.

მოთხოვნის შეტყობინება იხმარება, რათა მოითხოვოს კვანძისაგან მთლიანი სამარშრუტო ტაბულა ან მისი ნაწილი. მოთხოვნის შეტყობინებები ძირითადად გაიგზავნება ფართოსამაუწყებლო პაკეტების მეშვეობით. მოთხოვნის შეტყობინება მუშავდება ბიჯ-ბიჯად. ყოველ იტერაციაზე განიხილება შეტყობინების ერთი ჩანაწერი. თუ შეტყობინება შეიცავს მხოლოდ ერთ ჩანაწერს, რომლის მისამართის ოჯახის იდენტიფიკატორი 0-ის ტოლია, ხოლო მარშრუტის სიგრძის ველი - 16-ის, ეს არის მთლიანი სამარშრუტო ტაბულის მოთხოვნის შეტყობინება. მის საპასუხოდ გენერირდება პასუხი, რომელშიც ჩაიწერება მოცემული კვანძის სამარშრუტო ტაბულის ყველა ჩანაწერი. წინააღმდეგ შემთხვევაში შეტყობინება დამუშავდება შემდეგნაირად: ამოიკითხება დანიშნულების ქსელის მისამართი და მოიძებნება შესაბამისი ჩანაწერი სამარშრუტო ტაბულაში, ტაბულიდან ამოიკითხება მარშრუტის სიგრძის ველი და ჩაიწერება შეტყობინების შესაბამისი ჩანაწერის მარშრუტის სიგრძის ველში. თუ სამარშრუტო ტაბულაში შესაბამისი ჩანაწერი არ აღმოჩნდა, მაშინ შეტყობინების მარშრუტის სიგრძის ველში ჩაიწერება 16 (მარშრუტი არ არსებობს). შემდეგ ბრძანების ველში იწერება 2 (პასუხი) და მიღებული შეტყობინება იგზავნება იმ მისამართით საიდანაც მიღებულ იქნა მოთხოვნა.

თავის მხრივ პასუხის შეტყობინება შეიძლება გენერირებულ იქნას მოთხოვნის საპასუხოდ, პერიოდული განახლებისას და მარშრუტის ცვლილებით გამოწვეული განახლებისას (ტრიგერული განახლება). პასუხის მიღებისას ის პირველ რიგში უნდა შემოწმებულ იქნას, მისი წყაროს მისამართი თუ არის მეზობლების სიაში და თუ ძევს უშუალოდ მიერთებულ ქსელზე და წყაროს პორტი თუ არის 520. თუ ამ პირობებიდან რომელიმე არ სრულდება, პასუხი იგნორირებულ იქნება. გარდა ამისა, შემოწმდეს თუ წყაროს მისამართი ემთხვევა მოცემული კვანძის რომელიმე ინტერფეისის მისამართს. ამ შემთხვევაშიც ასეთი შეტყობინება უგულვებელყოფილ იქნება. ამის შემდეგ უნდა შემოწმდეს შეტყობინების ველების სისწორე. შემდეგი ეტაპია პასუხის ჩანაწერების თითო-თითოდ დამუშავება. გამოითვლება მიღებული მარშრუტის სიგრძე შეტყობინებაში მითითებული სიგრძის სიდიდეზე იმ ხაზის სიგრძის დამატებით, საიდანაც მოვიდა შეტყობინება.

```
metric = min[metric+cost,16]
```

თუ ეს მარშრუტი ჯერ არ არსებობს ტაბულაში და მისი სიგრძე არ არის 16, მაშინ ეს მარშრუტი შეიტანება ტაბულაში ახლად გამოთვლილი სიგრძით. ხდება ტაიმაუტის ტაიმერის ინიციალიზაცია, თუ მოცემული მარშრუტისათვის გაშვებული იყო გათიშვის ტაიმერი, ის ჩერდება და ყენდება ცვლილების ალამი. თუ ასეთი მარშრუტი ტაბულაში უკვე არსებობს, დარდება ერთმანეთს მარშრუტიზატორები საიდანაც მოვიდა მარშრუტი. თუ ისინი ემთხვევა ერთმანეთს და მარშრუტების სიგრძეები განსხვავებულია, ან თუ ისინი სხვადასხვაა და ახალი სიგრძე ნაკლებია ძველზე, ძველი მარშრუტი ჩანაცვლდება ახალი მარშრუტით. ამ შემთხვევაშიც ნულდება ტაიმაუტის ტაიმერი, ჩერდება გათიშვის ტაიმერი და ყენდება ცვლილების ალამი. თავის მხრივ თუ მარშრუტის ახალი სიგრძეა 16, გაიშვება გათიშვის ტაიმერი.

RIP პროტოკოლში პერიოდული განახლების შეტყობინებებში გადაიგზავნება მთლიანი სამარშრუტო ტაბულები, ხოლო სამარშრუტო ტაბულის ცვლილებით გამოწვეული განახლებებისას კი გადაიგზავნება მხოლოდ ის მარშრუტები, რომელთათვისაც დაყენებულია ცვლილების ალამი. ტრიგერული განახლების შემდეგ ცვლილების ალმები კვლავ იშლება.

გარდა ამისა RIP პროტოკოლს გააჩნია მასთან უშუალოდ მიერთებული ხაზების მწყობრიდან გამოსვლისა და მწყობრში ჩადგომის აღმოჩენის საშუალება, რომელიც ძირითადად ეფუძნება იმ პრინციპს, რომ თუ მოცემული ხაზით მიიღება დეიტაგრამები, მაშინ ხაზი ფუნქციონირებს.

ამ პროტოკოლში მწვავედ დგას "უსასრულობამდე თვლის" პრობლემა. მისი გადაწყვეტა აქ ხდება შედარებით მცირე მაქსიმალური მარშრუტის სიგრძის შემოღებით. თუ განვიხილავთ III თავში მოყვანილ მაგალითს, ალგორითმს დასჭირდება მხოლოდ 16 იტერაცია, სანამ ის მიხვდება, რომ გამოიყენოს ხაზი (2,1).

უსასრულობამდე თვლის თავიდან ასაცილებლად და პროტოკოლის კრებადობის გასაზრდელად გამოიყენება რამოდენიმე მეთოდი. ერთ-ერთი მათგანია "გაყოფილი ჰორიზონტი უკუ დახშობით" (Split horizon with poisoned reverse). ამ დროს რომელიმე მეზობლისაგან მიღებული მარშრუტი აღარ შეიტანება ამ მეზობლისაკენ გაგზავნილ განახლების შეტყობინებაში პირვანდელი სახით, არამეტ შეიტანება უსასრულობის (16) ტოლი სიგრძით. თუ კვანძი იმყოფება ფართოსამაუწყებლო ქსელზე, მაშინ ამგვარი შეტყობინებები გაეგზავნება ამ ქსელთან დაკავშირებულ ყველა მეზობელს. ამ მეთოდის გამოყენებისას, თუ რაიმე ორ კვანძს შორის არსებობს სამარშრუტო მარყუჟი, ის მაშინვე მოისპობა, მაგრამ განხილული მეთოდი საშუალებას არ იძლევა, აღმოჩენილ იქნას სამარშრუტო მარყუჟები სამ და მეტ კვან.ს შორის. სწორედ ამ პრობლემის გადაწყვეტას ემსაყურება ტრიგერული განახლებების მეთოდი. ამ მეთოდის მიხედვით თუ მოხდა სამარშრუტო ტაბულის რაიმე ცვლილება, მაშინვე მეზობლებისაკენ იგზავნება ტრიგერული განახლების შეტყობინებები მიუხედავად იმისა, არის თუ არა დრო პერიოდული განახლების შეტყობინების გაგზავნისა. გარდა ამისა, თუ რომელიმე კვანძი შეიცავს მარშრუტს, რაიმე მეზობლის გავლით, და მან მიიღო შეტყობინება ამ მარშრუტის სიგრძის ცვლილების შესახებ ამ მეზობლისაგან, მან უნდა შეიტანოს ეს ახალი სიგრძე თავის ტაბულაში მიუხედავად იმისა მეტია თუ ნაკლები ის ძველ მნიშვნელობაზე. მაგალითის სახით წარმოვიდგინოთ, რომ G მარშრუტიზატორიდან მოვიდა შეტყობინება N ქსელის მიუღწევლობის შესახებ და ძველი მარშრუტიც იყო G კვანძის გავლით, მაშინ უსასრულობის (16) ტოლი სიგრძე შეიტანება ტაბულაში და ტრიგერული განახლებები გაიგზავნება შემდგომ. ცვლილებები შეიტანება მხოლოდ

იმ კვანძების სამარშრუტო ტაბულებში, რომელთა მარშრუტები N ქსელამდე შეიცავს G კვანძს.

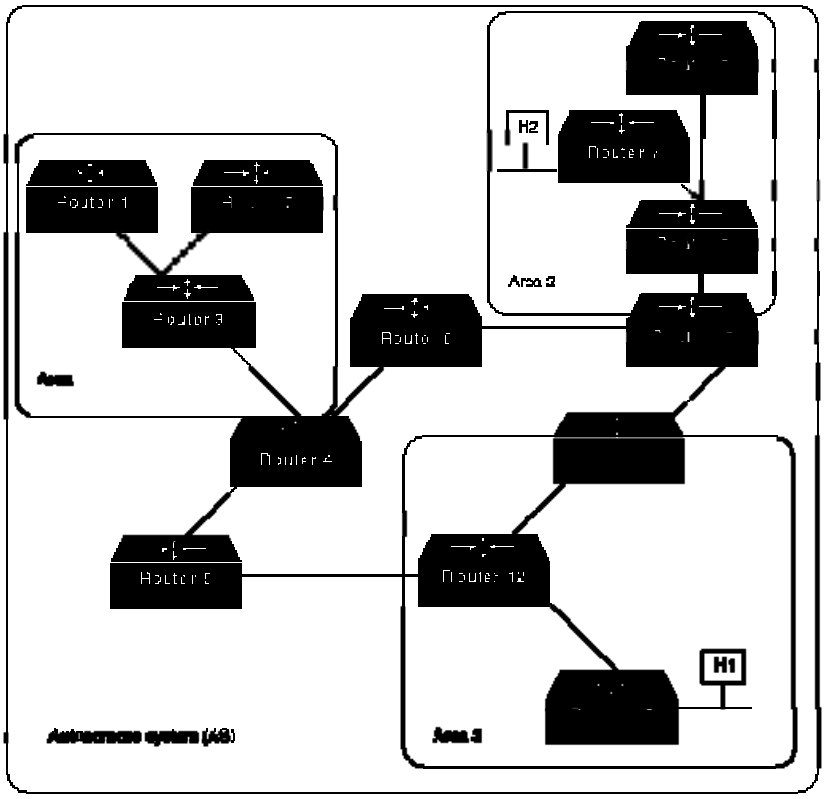
სამწუხაროდ ხშირია შემთხვევები, როცა პერიოდული განახლების შეტყობინებები კვლავ აღადგენენ მარშრუტების სიგრძეების შეცდომით მნიშვნელობებს. ეს პრობლემა წყდება შეკავების დროის (hold time) შემოღებით: მას შემდეგ რაც მიიღება ტრიგერული განახლების შეტყობინება და შესაბამისი ცვლილებები შეიტანება სამარშრუტო ტაბულაში, ახლად შეცვლილი ჩანაწერები გარკვეული დროით ბლოკირდება ნებისმიერი ცვლილებებისაგან.

ყველა მოყვანილი მეთოდი ზრდის მარშრუტიზაციის პროტოკოლის კრებადობას და განკუთვნილია სამარშრუტო კვანძებისა და სხვა ხელის შემშლელი მოვლენების თავიდან ასაცილებლად.

5.2. OSPF

OSPF პროტოკოლი შეიქმნა როგორც შიდა მარშრუტიზაციის პროტოკოლი, რომელსაც ეკისრება პაკეტების მარშრუტიზაცია ერთი ავტონომიური სისტემის ფარგლებში. პროტოკოლისაგან განსხვავებით, რომელიც წარმოადგენდა დისტანციურ-ვექტორულ (distance-vector) პროტოკოლს, ეს არის კავშირების მდგომარეობის (link state) პროტოკოლი. მასში შეიძლება თითოეული ხაზის სიგრძე დაუკავშირდეს ნებისმიერი კრიტერიუმების სიმრავლეს, მაგალითად, ხაზის დაყოვნება (მილიწამებში), ან გამტარუნარიანობა (ბიტ/წმ-ში). მასში უმოკლესი გზების გამოთვლა ხდება ქსელის ტოპოლოგიის შესახებ შეგროვებული მონაცემებზე დაყრდნობით დიკსტრას ალგორითმის გამოყენებით.

OSPF წარმოადგენს მარშრუტიზაციის იერარქიულ პროტოკოლს. ამ იერარქიის ყველაზე მაღალ დონეს ავტონომური სისტემა (AS) წარმოადგენს. ეს არის ერთიანი ადმინისტრირების ქვეშ მყოფი ქსელების ერთობლიობა რომლებიც ხმარობენ ერთიან მარშრუტიზაციის პროტოკოლს. ერთი ავტონომიური სისტემის შიგნით ქსელები განაწილებული ცალკეულ უბნებად. ერთი უბნის ყველა კვანძს გააჩნია ერთნაირი მონაცემები ამ უბნის ტოპოლოგიის შესახებ, რომელიც ინახება ტოპოლოგიურ მონაცემთა ბაზაში. სწორედ ამ მონაცემების მიხედვით გამოითვლება უმოკლესი მარშრუტები და შედგება სამარშრუტო ტაბულა. ტოპოლოგიურ მონაცემთა ბაზაში ქსელის უბანი წარმოდგენილია ორიენტირებული გრაფის სახით, სადაც გრაფის წვეროებია მარშრუტიზატორები, ხოლო რკალები შესაბამისად ხაზები, რომელთაც მიეწერებათ გარკვეული სიგრძეები. უბნები ერთმანეთთან არიან დაკავშირებულნი მაგისტრალის საშუალებით, რომელიც თავის მხრივ აგრეთვე წარმოადგენს უბანს. მაგისტრალი შესაძლოა არ იყოს ბმული, მაშინ საჭიროა შეიქმნას ვირტუალური კავშირი იმ უბანში, რომელიც აერთებს მაგისტრალის ორ არაბმულ ნაწილს. თითოეულმა უბანმა არ იცის სხვა უბნებისა და მაგისტრალის ტოპოლოგია. პროტოკოლის იერარქია მოცემულია შემდეგ სურათზე:



აქედან გამომდინარე პროტკოლში არსებობს 3 სახის მარშრუტიზატორი:

- უბნის შიდა მარშრუტიზატორი, რომლის ყველა ინტერფეისი ეკუთვნის ერთ და იგივე უბანს. მასში მოქმედებს ალგორითმის ერთი ასლი და გააჩნია ერთი ტოპოლოგიური მონაცემთა ბაზა;
- უბნის კიდური მარშრუტიზატორი, რომლის ერთ-ერთი ინტერფეისი ეკუთვნის განსხვავებულ უბანს ან მაგისტრალს. მასში მუშაობს ალგორითმის იმდენი ასლი, რამდენი უბანიცაა მასთან მიერთებული;
- სასაზღვრო მარშრუტიზატორი, რომელიც მიმოცვლის სამარშრუტო ინფორმაციას სხვა ავტონომიური სისტემების მარშრუტიზატორბთან.

შესაბამისად არსებობს მარშრუტიზაციის 3 სახე:

- უბნის შიდა მარშრუტიზაცია, როცა პაკეტის წყაროც და მიმღებიც იმყოფება ერთ და იგივე უბანში და ამგვარად ის ამ უბანს არ ტოვებს.
- უბანთაშორისი მარშრუტიზაცია, როცა პაკეტის წყარო და მიმღები იმყოფებიან სხვადასხვა უბანში. ამ შემთხვევაში მთელი მარშრუტი იყოფა სამ ნაწილად: წყაროდან წყაროს უბნის კიდურ მარშრუტიზატორამდე, წყაროს და მიმღების უბნების კიდურ მარშრუტიზატორებს შორის მაგისტრალის გავლით და მიმღების უბნის კიდური მარშრუტიზატორიდან თვით მიმღებამდე. თითოეული ნაწილის ფარგლებში ადგილი აქვს უბნის შიდა მარშრუტიზაციას.
- გარე მარშრუტიზაცია, როცა მიმღები იმყოფება ავტონომიური სისტემის ფარგლებს გარეთ. ეს მარშრუტები მიიღება გარე მარშრუტიზაციის პროტოკოლისაგან (როგორიცაა BGP) ან სტატიკური მარშრუტების საშუალებით შეიტანება ადმინისტრატორის მიერ. არსებობს 2 სახის გარე მარშრუტი: I ტიპის გარე მარშრუტის სიგრძე შესადარია OSPF-ის შიდა მარშრუტების სიგრძეებთან. II ტიპის გარე მარშრუტების სიგრძის კრიტერიუმი მნიშვნელოვნად განსხვავდება

ავტონომიური სისტემის შიდა კრიტერიუმისაგან. ამ შემთხვევაში ყველა ასეთი გარე მარშრუტის სიგრძე მეტია, ვიდრე ნებისმიერი შიდა მარშრუტის სიგრძე. ყოველი გარე მარშრუტი ინიშნება 32 ბიტიანი რიცხვით, რომელიც აღნიშნავს ავტონომიურ სისტემას, საიდანაც იყო მიღებული მარშრუტი.

პროტოკოლი იძლევა საშუალებას ერთდროულად გამოყენებულ იქნას რამოდენიმე ტოლი სიგრძის მქონე მარშრუტი, რაც მნიშვნელოვნად ზრდის ქსელის გამტარუნარიანობას.

OSPF პროტოკოლის ერთ-ერთ ძირითად ნაწილს წარმოადგენს SPF ალგორითმი, რომელიც ეფუძნება დიკსტრას უმოკლესი გზის ალგორითმს. ყოველ მარშრუტიზატორში მუშაობს ალგორითმის იმდენი ასლი, რამდენი უბანიცაა მასთან მიერთებული. კვების მიწოდებისთანავე მარშრუტიზატორი ახდენს თავისი მონაცემთა სტრუქტურების ინიციალიზაციას და ქვემდგომი დონის პროტოკოლების საშუალებით არკვევს თავისი ინტერფეისების მდგომარეობას.

მას შემდეგ რაც მარშრუტიზატორი დარწმუნდება თავისი ინტერფეისების მუშა მდგომარეობაში, ის იყენებს Hello პროტოკოლს მეზობლების მოპოვებისათვის. გარდა მეზობლების მოპოვების ფუნქციისა ეს პროტოკოლი გამოიყენება სხვა მარშრუტიზატორებისათვის შესატყობინებლად, რომ მოცემული მარშრუტიზატორი ფუნქციონირებს. ფართოსამაუწყებლო და წერტილი-წერტილი ტიპის ქსელებში პაკეტები ავტომატურად ეგზავნება ამ ქსელის ყველა მარშრუტიზატორს, რაც შეეხება არაფართოსამაუწყებლო ტიპის ქსელებს, როგორიცაა Frame Relay და ATM, საჭიროა ქსელის ადმინისტრატორის ჩარევა, რათა მიუთითოს მოცემული მარშრუტიზატორის მეზობლები.

ამის შემდეგ ხდება ყოველ ქსელში მოსაზღვრეობების ჩამოყალიბება. მოსაზღვრეა ორი მარშრუტიზატორი, რომელთა ტოპოლოგიური მონაცემთა ბაზები სინქრონიზირებულია. მოსაზღვრეობები აკონტროლებენ მარშრუტიზაციის პროტოკოლის და ტოპოლოგიური მონაცემთა ბაზის განახლებების გავრცელებას. მარშრუტიზატორი პერიოდულად უგზავნის ტოპოლოგიური ბაზის განახლებების შეტყობინებებს თავის მოსაზღვრე მარშრუტიზატორებს. ამგვარად ეს შეტყობინებები გაჭოლავენ მთელს უბანს და უზრუველყოფენ, რომ უბნის ყოველ კვანძში ინახება ერთი და იგივე ტოპოლოგიური მონაცემთა ბაზა.

გარდა ზემოთ ხსენებულისა, Hello პროტკოლის მეშვეობით ყოველ ქსელში ხდება ე.წ. მიკუთვნილი მარშრუტიზატორის (designated router) არჩევა. მიკუთვნილ მარშრუტიზატორს აკისრია ორი მნიშვნელოვანი ფუნქცია: იმ ქსელისათვის, რომელშიც იგი არის მიკუთვნილი, ის აგზავნის ქსელის კავშირების შეტყობინებებს (network link state advertisement), რომელშიც მითითებულია ამ ქსელთან დაკავშირებული ყველა მარშრუტიზატორი და იგი ხდება ამ ქსელის ყველა მარშრუტიზატორის მოსაზღვრე და ამით ცენტრალურ ადგილს იკავებს მთელი ქსელის ტოპოლოგიური ბაზების სინქრონიზაციაში.

მიკუთვნილი მარშრუტიზატორის არჩევა შემდეგნაირად ხდება: როგორც კი მარშრუტიზატორის ინტერფეისი იწყებს ფუნქციონირებას, ის ამოწმებს ამ ქსელში თუ არის უკვე მიკუთვნილი მარშრუტიზატორი. თუ არის, ინტერფეისი ეთანხმება ამ ფაქტს მიუხედავად მისი პრიორიტეტისა. თუ მიკუთვნილი მარშრუტიზატორი არ არსებობს, მაშინ მარშრუტიზატორი, რომელსაც ეკუთვნის ეს ინტერფეისი ნიშნავს თავის თავს მიკუთვნილად. თუ რამოდენიმე მარშრუტიზატორი გახდა ერთდროილად მიკუთვნილი, მაშინ მიკუთვნილად რჩება უფრო მაღალი

პრიორიტეტის მქონე. პრიორიტეტების ტოლობის შემთხვევაში აირჩევა უფრო დიდი ID-ს მქონე მარშრუტიზატორი.

მას შემდეგ, რაც მარშრუტიზატორი დააგროვებს მონაცემებს უბნის ტოპოლოგიის შესახებ, მიღებული ტოპოლოგიური მონაცემთა ბაზის მიხედვით ის ადგენს უბნის უმოკლესი გზების ხეს, რომლის ფესვს თვითონ წარმოადგენს. ამ ხის საფუძველზე იქმნება სამარშრუტო ტაბულა.

პროტოკოლის ყველა პაკეტი იწყება სტანდარტული 24 ბაიტიანი სათაურით:

სათაურის პირველი ველი განსაზღვრავს პროტკოლის ვერსიას და გამოიყენება თავსებადობის პრობლემების თავიდან ასაცილებლად. აღსანიშნავია, რომ ამ მომენტისათვის ხმარებაშია OSPF 2, რომელიც არ არის თავსებადი 1 ვერსიის OSPF-თან.

შემდეგი ველი განსაზღვრავს პაკეტის ტიპს. არსებობს 5 ტიპის პაკეტი:

- ტიპი = 1 - Hello პროტკოლის პაკეტი. ასეთი პაკეტები პერიოდულად გადაიგზავნება მეზობლური ურთიერთობის დასამყარებლად და უზრუნველსაყოფად. ამ პაკეტების საშუალებით მარშრუტიზატორებს შორის თანხმდება სხვადასხვა პარამეტრები, როგორიცაა ქსელის ნიღბები, მისალმების პერიოდები, მარშრუტიზატორების პრიორიტეტები და კავშირის დაკარგვის სიგნალიზაცია.
- ტიპი = 2 - მონაცემთა ბაზის აღწერა (database description). ეს პაკეტები აღწერენ ტოპოლოგიური მონაცემთა ბაზის შიგთავსს. ამ პაკეტების გადაგზავნა ხდება მოსაზღვრე მარშრუტიზატორების ინტერფეისების ინიციალიზაციისას.
- ტიპი = 3 - არხის მდგომარეობის მოთხოვნა (link state request). ამ პაკეტებით მიმოცვლას აქვს ადგილი, როცა რომელიმე მარშრუტიზატორი აღმოაჩენს, რომ მისი ტოპოლოგიური მონაცემთა ბაზის ნაწილი მოძველდა.
- ტიპი = 4 - არხის მდგომარეობის კორექტირება (link state update). ეს შეტყობინებები გაიგზავნება კავშირის მდგომარეობის მოთხოვნის საპასუხოდ. ამ პაკეტების საშუალებით ვრცელდება ქსელის შესახებ სხვადასხვა ინფორმაცია. კავშირის მდგომარეობის კორექტირება შეიძლება შეიცავდეს შემდეგი ტიპის მონაცემებს:

  ⇨ მარშრუტიზატორის არხების შეტყობინებები (router links advertisements). ყოველი მარშრუტიზატორი აგზავნის ამ შეტყობინებებს, სადაც მოცემულია ყველა არხის მდგომარეობა, რომლებიც აკავშირებენ მას რაღაც გარკვეულ

უბანთან. ეს შეტყობინებები გაჭოლავენ მთელ უბანს, მაგრამ არ სტოვებენ მის ფარგლებს.

⇨ ქსელის არხების შეტყობინებები (network link advertisements). ეს შეტყობინებები აღწერენ მოცემულ ქსელთან მიერთებულ ყველა მარშრუტიზატორს. ისინი გადაიცემა ამ ქსელის მიკუთვნილი მარშრუტიზატორის მიერ და ვრცელდება მთელს უბანში, რომელიც შეიცავს ამ ქსელს, მაგრამ არ გადის მის გარეთ.

⇨ ჯამური არხების შეტყობინებები (summary links advertisements). ეს შეტყობინებები აჯამავენ მარშრუტებს რაიმე დანიშნულების ადგილამდე, რომელიც ძევს ამ უბნის გარეთ, მაგრამ მოცემული ავტონომიური სისტემის შიგნით. ასეთი პაკეტები გენერირდება უბნის კიდური მარშრუტიზატორების მიერ. მაგისტრალში იგზავნება შეტყობინებები მხოლოდ უბნების შიდა მარშრუტიზატორების შესახებ, ხოლო ცალკეული უბნების შიგნით კი - შეტყობინებები როგორც უბნის შიდა, ისე უბანთაშორისი მარშრუტიზატორების შორის შესახებ.

⇨ ავტონომიური სისტემის გარე არხების შეტყობინებები (AS external links advertisements). ეს შეტყობინება აღწერს მარშრუტს, რომლის დანიშნულების ადგილი ძევს მოცემული ავტონომიური სისტემის ფარგლებს გარეთ. ის გენერირდება ავტონომიური სისტემის სასაზღვრო მარშრუტიზატორით, რომელიც თავის მხრივ იღებს მას გარე მარშრუტიზაციის პროტკოლისაგან (მაგალითად BGP-საგან). ამ სახის შეტყობინებები გაჭოლავს მთელ ავტონომიურ სისტემას მიუხედავად უბნების საზღვრებისა.

- ტიპი = 5 - არხის მდგომარეობის დადასტურება (link state acknowledgement). ამ ტიპის პაკეტი ადასტურებს არხის მდგომარეობის კორექტირების პაკეტის მიღების ფაქტს.

პაკეტის ტიპის ველს მოყვება პაკეტის სიგრძის ველი (2 ბაიტი), რომელიც განსაზღვრავს პაკეტის მთლიან სიგრძეს ბაიტებში სათაურის ჩათვლით.

მარშრუტიზატორის ID განსაზღვრავს პაკეტის წყაროს და ძირითადად ემთხვევა პაკეტის გამგზავნი მარშრუტიზატორის IP მისამართს.

უბნის ID აიდენტიფიცირებს უბანს, რომელსაც ეკუთვნის მოცემული პაკეტი.

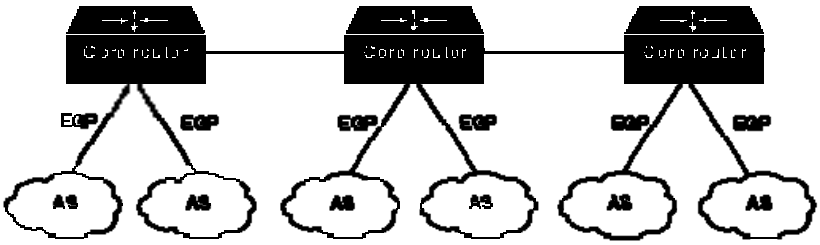
საკონტროლო ჯამი წარმოადგენს 16 ბიტიან ველს, რომლის დანიშნულებაა შეცდომების კონტროლი. ის იძლევა საშუალებას აღმოჩენილ იქნას პაკეტის გადაცემისას მონაცემთა დამახინჯება.

ამის შემდეგ პაკეტში არის აუტენტიკაციის ტიპის 2 ბაიტიანი ველი, რომელსაც მოჰყვება აუტენტიკაციის მონაცემები (64 ბიტი). ამ ველების საშუალებით პაკეტის წყარო ახდენს თავისი თავის იდენტიფიცირებას მიმღებთან ურთიერთობისას. ეს საშუალება იძლევა დამატებით უშიშროებას უზრუნველყოფნას და თავიდან აცილებს არასასურველი ინფორმაციის შემოჭრას სამარშრუტო ტაბულებში.

OSPF პროტოკოლს გააჩნია სხვადასხვა ზედა დონის სერვისის განსხვავების საშუალება (TOS - Type Of Service). მას შეუძლია იმისდა მიხედვით თუ რა სერვისს ეკუთვნის მონაცემთა პაკეტი გადააგზავნოს ის მაღალსიჩქარიანი ან სანდო მაღალპრიორიტეტული მარშრუტებით და ამგვარად მიაღწიოს მთლიანი ქსელის გამტარუნარიანობის უფრო ეფექტურ გამოყენებას.

5.3. EGP

EGP მიეკუთვნება გარე მარშრუტიზაციის პროტოკოლების ოჯახს, რომლებიც ახდენენ მარშრუტების მიმოცვლას სხვადასხვა ავტონომიურ სისტემებს შორის. ეს პროტკოლი შეიქმნა იმ დროს, როცა ინტერნეტი ჯერ განვითარების პროცესში იყო და ამიტომ ხასიათდება სიმარტივით. EGP პროტკოლი მოქმედებს ავტონომიური სისტემების სასაზღვრო მარშრუტიზატორებზე. EGP-ს არ გააჩნია მარშრუტების გამოთვლის დამოუკიდებელი ალგორითმი. ის მხოლოდ მიმოცვლის მონაცემებს სასაზღვრო მარშრუტიზატორებს შორის. EGP ქსელს შემდეგი სტრუქტურა გააჩნია:



იმის მიუხედავად, რომ EGP წარმოადგენს დინამიური მარშრუტიზაციის პროტკოლს, მისი მუშაობის სქემა მეტად მარტივია. მას არ შეუძლია ინტელექტუალური გადაწყვეტილებების მიღება მარშრუტიზაციის შესახებ. მისი შეტყობინებები შეიცავენ ინფორმაციას მხოლოდ ქსელების მიღწევადობის შესახებ, ე.ი. მითითებას, რომ გარკვეულ ქსელებში საინფორმაციო პაკეტები შეიძლება მოხვდნენ გარკვეული მარშრუტიზატორების გავლით.

EGP ასრულებს სამ ძირითად ფუნქციას:

- EGP მარშრუტიზატორები მუშაობისას ქმნიან მეზობელთა სიმრავლეს, რომლებთან ერთად კოლექტიურად ხმარობენ ინფორმაციას ქსელების მიღწევადობის შესახებ.
- EGP მარშრუტიზატორები პერიოდულად არკვევენ მეზობელთა ფუნქციონირების საკითხს.
- EGP მარშრუტიზატორები აგზავნიან კორექტირების შეტყობინებებს თავის ავტონომიურ სისტემის ფარგლებში ქსელების მიღწევადობის შესახებ.

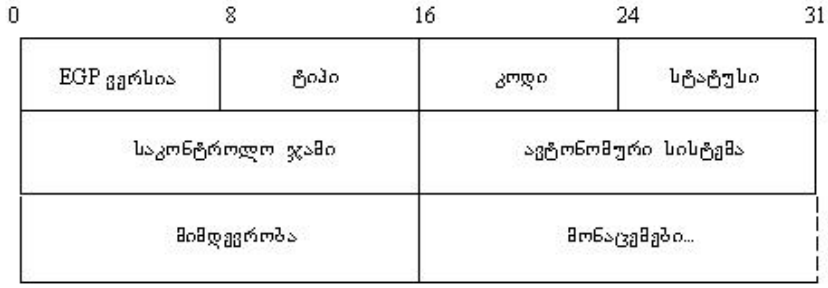
ამ ფუნქციების რეალიზაციისათვის პროტკოლი იყენებს შემდეგ შეტყობინებების სისტემას:

- ⇨ მეზობლის აღმოჩენა (neighbor acquisition). ამ პაკეტების საშუალებით ყოველი მარშრუტიზატორი აღმოაჩენს თავის მეზობლებს და ათანხმებს შეტყობინებათა მიმოცვლის პარამეტრებს, როგორიცაა: მისალმების ინტერვალი (hello interval) და გამოკითხვის ინტერვალი (poll interval). მისალმების ინტერვალი განსაზღვრავს მეზობელთა ფუნქციონირების შემოწმების პერიოდს, ხოლო გამოკითხვის ინტერვალი – მარშრუტების კორექტირების პერიოდს. მეზობლის აღმოჩენა ხდება ე.წ. სამსვლიანი კვიტირების გამოყენებით (ე.ი. მოთხოვნა / პასუხი / უარი).
- ⇨ მეზობლის მიღწევადობა (neighbor reachability). ეს შეტყობინება ატყობინებს მეზობლებს მოცემული მარშრუტიზატორის ფუნქციონირების შესახებ. თუ მარშრუტიზატორი არ იღებს ამ შეტყობინების რაღაც პროცენტს, იგი თვლის,

რომ ეს მეზობელი გამოვიდა მწყობრიდან. ამ ტიპს ეკუთვნის ორი სახის შეტყობინება: მისალმება (hello) და მისალმებაზე პასუხი (I hear you).

- გამოკითხვა (poll). სწორი მარშრუტიზაციისათვის მარშრუტიზატორმა უნდა იცოდეს, თუ რომელი ქსელის მიმართ ხდება ცალკეული სადგურების ადგილმდებარეობის განსაზღვრა. ეს შეტყობინება შეიცავს ამ ქსელის IP მისამართის დამატებით ველს.
- მარშრუტიზაციის კორექცია (routing update). ამ შეტყობინებების საშუალებით მარშრუტიზატორი ატყობინებს თავის მეზობლებს მის ავტონომიური სისტემის შიგნით არსებული ქსელების შესახებ. იგი შეიცავს რამოდენიმე დამატებით ველს: შიდა მარშრუტიზატორების რიცხვი, გარე მარშრუტიზატორების რიცხვი, შეტყობინების წყაროს ქსელის მისამართი. ამას მოსდევს ამ მარშრუტზე მყოფი მარშრუტიზატორების ჩამონათვალი და ამ მარშრუტიზატორებთან მიერთებულ ქსლებამდე მანძილები. EGP პროტოკოლი არ იყენებს ამ მანძილებს, ამიტომ ის წარმოადგენს მხოლოდ მიღწევადობის პროტკოლს.
- შეცდომები (error). ეს შეტყობინება გაიგზავნება, თუ მოხდა რაიმე პროტკოლით გაუთვალისწინებელი შემთხვევა. ის შეიცავს შემდეგ დამატებით ველებს: მიზეზი (reason) და შეცდომის სათაური (message header). შეცდომის შეტყობინება შეიძლება წარმოიშვას შემდეგი მიზეზების გამო: პაკეტის სათაურის არასწორი ფორმატი, პაკეტის მონაცემთა ველის არასწორი ფორმატი და ა.შ.

EGP შეტყობინების პაკეტის ფორმატი შემდეგია:



პაკეტის პირველი ველი აღნიშნავს EGP პროტოკოლის ვერსიას. ეს ველი გამოიყენება, რათა შემოწმდეს პროტკოლთა ვერსიების შესაბამისობა პაკეტის წყაროსა და მიმღებს შორის.

პაკეტის ტიპის ველი განსაზღვრავს ზემოთ აღნიშნულიდან ერთ-ერთს. მისი მნიშვნელობაზე არის დამოკიდებული მონაცემთა ველის შიგთავსი. კოდის ველი კი განსაზღვრავს ზემოთ მოყვანილი ტიპების ქვეტიპებს.

სტატუსის ველი დამოკიდებულია შეტყობინების ტიპზე და აღნიშნავს პროტკოლის მუშაობის მდგომარეობას. ის შეიძლება აღნიშნავდეს: რესურსების უკმარისობას, არასწორ პარამეტრებს, პროტკოლის დარღვევებს და ა.შ.

საკონტროლო ჯამის ველი გამოიყენება პაკეტის გადაცემისას შეცდომების აღმოსაჩენად. შეცდომის აღმოჩენის შემთხვევაში გენერირდება შესაბამისი შეცდომის შეტყობინება.

ავტონომიური სისტემის ნომერი ცალსახად აიდენტიფიცირებს იმ ავტონომიურ სისტემას, რომელსაც ეკუთვნის პაკეტის წყარო.

მიმდევრობის ნომერი იძლევა საშუალებას ყოველ მოთხოვნის პაკეტს შეუსაბამდეს პასუხის პაკეტი. ახალ მეზობელთან კავშირის დამყარებისას მიმდევრობის ნომერი დგება საწყის ნულოვან მდგომარეობაში და ყოველი მომდევნო მოთხოვნა/პასუხის შემდეგ იზრდება ერთით.

მიმდევრობის ნომერს მოჰყვება მონაცემთა ცვლადი ველი, რომელიც შეიცავს პაკეტის ტიპზე დამოკიდებულ დამატებით პარამეტრებს.

5.4. BGP

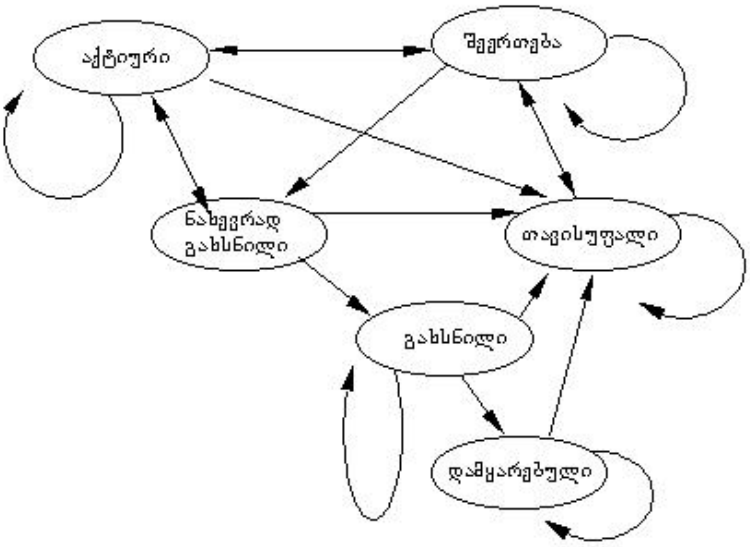
EGP პროტკოლისაგან განსხვავებით BGP წარმოადგენს უფრო რთულ პროტოკოლს. იგი ეკუთვნის იგივე გარე მარშრუტიზაციის პროტოკოლების ოჯახს, თუმცა მას შეუძლია მოახდინოს მარშრუტიზაცია ავტონომიური სისტემის შიგნითაც. მონაცემთა გადაცემისას ის ეყრდნობა TCP პროტკოლს და ინფორმაციის მიმოცვლისათვის იყენებს 179 პორტს.

BGP არ გამოიყენება უშუალოდ საინფორმაციო პაკეტების მარშრუტიზაციისათვის. მისი ძირითადი დანიშნულებაა სამარშრუტო ინფორმაციის მიმოცვლა ავტონომიურ სისტემებს შორის, ამ ინფორმაციის ანალიზი და მარშუტების ინჯექტირება ავტონომიური სისტემის შიდა მარშრუტიზაციის პროტოკოლში (IGP). ოფიციალურ პროტოკოლის აღწერაში მითითებულია, რომ მეზობელი მარშრუტიზატორები აუცილებლად უნდა იყონ დაკავშირებულნი ერთ ფიზიკურ ქსელთან, მაგრამ BGP პროტოკოლის Cisco-ს რეალიზაციაში გათვალისწინებულია ე.წ. Multihop BGP, რომელიც ხსნის ამ შეზღუდვას. ცხადია, ამ შემთხვევაში BGP მეზობლებს შორის შეტყობინებების პაკეტების გადაცემა უნდა ეყრდნობოდეს რაიმე სხვა მარშრუტიზაციის პროტოკოლს (თუნდაც სტატიკურ მარშრუტიზაციას).

ორი BGP მარშრუტიზატორი სამეზობლო ურთიერთობის დამყარებისას ხსნიან TCP კავშირს, ათანხმებენ პარამეტრებს და ურთიერთგაცვლიან თავიანთ სამარშრუტო ტაბულებს. სამარშრუტო ტაბულების შემდგომ კორექტირებას არ აქვთ პერიოდული ხასიათი. ისინი გენერირდებიან მხოლოდ სამარშრუტო სქემის შეცვლის შემთხვევაში. რადგან მარშრუტის კორექტირების შეტყობინებაში შეიტანება ყველა შუალდური ავტონომიური სისტემა, ადვილი ხდება სამარშრუტო მარყუჟების აღმოჩენა და თავიდან აცილება.

მეზობლები პერიოდულად უგზავნიან ერთმანეთს შეტყობინებებს თავიანთი მუშა მდგომარეობის შესახებ. სხვადასხვა შეცდომების აღმოჩენისას გაიგზავნება შესაბამისი შეცდომის შეტყობინებები. ერთი ავტონომიური სისტემის საზღვარზე მყოფი BGP მარშრუტიზატორებიც ცვლიან ინფორმაციას ერთმანეთს შორის, რათა შექმნან ერთიანი წარმოდგენა ავტონომიური სისტემის შიდა სტრუქტურის შესახებ და გადაწყდეს, თუ რომელი სასაზღვრო მარშრუტიზატორი ჩაითვალოს შეერთების წერტილად პაკეტების მიღება-გადაცემისას. მიუხედავად იმისა, რომ BGP სამარშრუტო ტაბულაში ინახავს რაიმე დანიშნულების ადგილამდე ყველა არსებულ მარშრუტს, კორექტირების შეტყობინებებით გადაიცემა ყოველი დანიშნულების ადგილისათვის მხოლოდ ერთი - ოპტიმალური მარშრუტი. მარშრუტის ოპტიმალობის განსაზღვრა ბაზირდება კრიტერიუმების გარკვეულ სიმრავლეზე.

შემდეგი სქემა დეტალურად აღწერს BGP პროტკოლის მუშაობის თანამიმდევრობას:

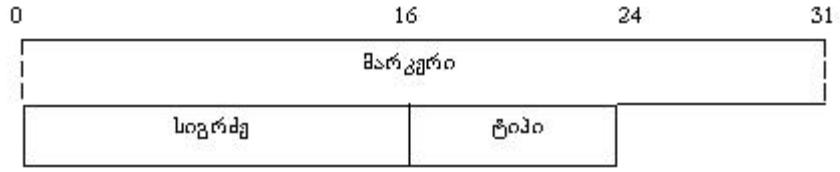


პროტკოლი გაივლის შემდეგ სტადიებს:

- თავისუფალი მდგომარეობა არის პირველი მდგომარეობა, რომელიც გრძელდება, სანამ არ გაიშვება BGP პროტკოლი ადმინისტრატორის მიერ. ამის შემდეგ ხდება რესურსების, შეერთების მცდელობების მთვლელის და TCP პროტკოლის ინიციალიზაცია და კვანძი უსმენს შესაძლო შეერთებებს მოცილებული კვანძებიდან. ამის შემდეგ იგი გადადის შეერთების მდგომარეობაში, შეცდომების შემთხვევაში კი ისევ ბრუნდება თავისუფალ მდგომარეობაში.
- შეერთების მდგომარეობაში კვანძი ელოდება სატრანსპორტო დონის შეერთების ₩არმატებით დასრულებას. თუ კავშირი ₩არმატებით დამყარდა, კვანძი აგზავნის გამხსნელ შეტყობინებას და გადადის ნახებრად გახსნილ მდგომარეობაში. თუ შეერთება ვერ განხორციელდა ₩არმატებით, კვანძი გადადის აქტიურ მდგომარეობაში. შეერთებების მცდელობების მთვლელის ამოწურვის შემთხვევაში შეერთების მდგომარეობა არ იცვლება და ხდება სატრანსპორტო პროტოკოლის ხელახალი ინიციალიზაცია.
- აქტიურ მდგომარეობაში კვანძი ცდილობს გაიჩინოს მეზობელი სხვა კვანძთან კავშირის დამყარებით. თუ კავშირი ₩არმატებით დამყარდა, ის გადადის ნახევრად გახსნილ მდგომარეობაში. თუ შეერთებების მთვლელი ამოიწუა, ხდება მისი ხელახალი ინიციაიალიზაცია და კვანძი ბრუნდება შეერთების მდგომარეობაში. ამასთან ერთად კვანძი უსმენს შესაძლო შეერთებებს სხვა კვანძებისაგან. თუ მდგომარეობები მუდმივად იცვლება აქტიურიდან შეერთებაზე და პირიქით, ეს ნიშნავს, რომ რაღაც პრობლემებია სატრანსპორტო პროტოკოლის ფუნქციონირებასთან დაკავშირებით.
- ნახევრად ღია მდგომარეობაში კვანძი ელოდება გამხსნელ შეტყობინებას მის მეზობლისაგან, ამოწმებს ამ შეტყობინების სისწორეს და ₩არმატების შემთხვევაში აინიციალიზირებს არსებობის ტაიმერს და იწყებს არსებობის შეტყობინებების გაგზავნას. შეცდომის აღმოჩენის შემთხვევაში გაიგზავნება შეცდომის შეტყობინება და კვანძის მდგომარეობა გახდება თავისუფალი. იგივეს ექნება ადგილი, თუ TCP კავშირი რაიმე მიზეზების გამო დაირღვევა.

- გახსნილ მდგომარეობაში კვანძი ელოდება მეზობლისაგან არსებობის ან შეცდომის შეტყობინებას. თუ მან მიიღო არსებობის შეტყობინება, კვანძი იცვლის მდგომარეობას დამყარებულზე, რაც ნიშნავს. რომ სამეზობლო კავშირი დამყარებულია. თუ კვანძი მიიღებს არსებობის ან კორექტირების შეტყობინებებს, შეკავების ტაიმერი ნულდება. შეცდომის შეტყობინების მიღების შემთხვევაში კვანძი კვლავ ბრუნდება თავისუფალ მდგომარეობაში.
- დამყარებული მდგომარეობა წარმოადგენს სამეზობლო კავშირის დამყარების ბოლო სტადიას. ამ სტადიაზე კვანძი იწყებს კორექტირების შეტყობინებების მიმოცვლას თავის მეზობლებთან. ყოველის არსებობის ან კორექციის შეტყობინების მიღებისას შეკავების ტაიმერი თავიდან იშვება, ხოლო შეცდომის არმოჩენის შემთხვევაში კვანძი ბრუნდება თავისუფალ მდგომარეობაში.

BGP შეტყობინების სტანდარტულ სათაურს აქვს შემდეგი სახე:



მარკერის ველი შეიცავს სიდიდეს, რომლის პროგნოზირება შეუძლია მიმღებს. იგი გამოიყენება შეტყობინების იდენტიფიცირებისათვის ან ორ მეზობელს შორის დაკარგული სინქრონიზაციის აღსადგენად.

სიგრძის ველი შეიცავს მთლიანი პაკეტის სიგრძეს ბაიტებში სათაურის და მონაცემების ჩათვლით.

ტიპის ველი განსაზღვრავს, თუ რა ფუნქციას ასრულებს პაკეტი მარშრუტიზაციის პროტოკოლის მუშაობის პროცესში. მისი მნიშვნელობები შემდგომ იქნება დეტალურად განხილული.

სტანდარტულ სათაურს პაკეტში მოყვება ცვლადი სიგრძის მონაცემები, რომელთა შიგთავსი განისაზღვრება პაკეტის ტიპის ველით.

BGP მარშრუტიზატორები მონაცემთა მიმოცვლისას იყენებენ შეტყობინებების ტიპების შემდეგ სისტემას:

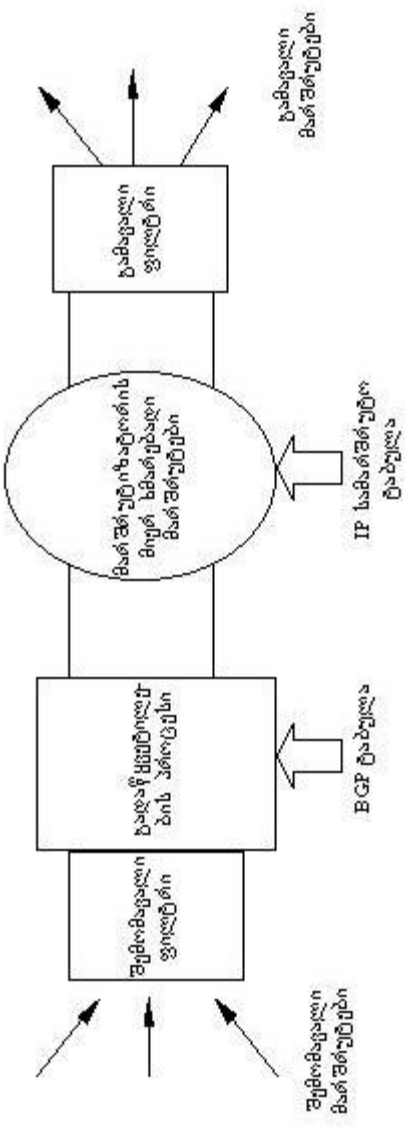
- გამხსნელი შეტყობინება (open message). მას შემდეგ რაც დამყარდება TCP კავშირი ორ მეზობელს შორის, ყოველი მხარე პირველ რიგში აგზავნის ამ შეტყობინებას, რომელიც შეიცავს მონაცემთა ველში რამოდენიმე დამატებით პარამეტრს: ვერსიის ნომერი, წყაროს ავტონომიური სისტემის ნომერი, შეკავების დრო, რომელიც განსაზღვრავს, თუ რამდენი წამის შემდეგ შეტყობინებების არმიღების შემთხვევაში ჩაითვალოს მეზობელი მარშრუტიზატორი მწყობრიდან გამოსულად. გარდა ამისა გადაიგზავნება აუთენტიკაციისათვის საჭირო ინფორმაცია.
- კორექტირების შეტყობინება (update message). ეს შეტყობინება გამოიყენება სხვა BGP სისტემების სამარშრუტო ტაბულების განახლებისათვის. ამ შეტყობინებით მიღებული ინფორმაციის საშუალებით აიგება ავტონომიური სისტემების ურთიერთკავშირების გრაფი. ამ შეტყობინებას გააჩნია 5 სახის დამატებითი პარამეტრი:

⇨ წყარო (origin). ეს პარამეტრი შეიძლება იღებდეს მნიშვნელობებს IGP, EGP და incomplete. IGP ნიშნავს, რომ ეს ქსელი წარმოადგენს მოცემული ავტონომიური სისტემის ნაწილს, EGP აღნიშნავს, რომ მოცემული მარშრუტი მოეწოდა მარშრუტიზატორს ავტონომიური სისტემის გარედან გარე მარშრუტიზაციის პროტოკოლის მეშვეობით. incomplete იხმარება იმის აღსანიშნავად, რომ ამ ქსელის შესახებ ცნობილი გახდა რაიმე სხვა საშუალებებით.
⇨ გზა (path), რომელიც განსაზღვრავს ავტონომიურ სისტემებს, რომლებიც მდებარეობენ გზაზე დანიშნულების ადგილამდე.
⇨ შემდეგი გადაგზავნა (next hop), წარმოადგენს მარშრუტიზატორის მისამართს, რომლისკენაც უნდა გადაიგზავნოს პაკეტი იმ ქსელების მისაღწევად, რომლებიც ჩამოთვლილია შეტყობინებაში.
⇨ მიუღწევადი (unreachable). ამ პარამეტრში ჩამოთვლილია მარშრუტები რომლებიც გამოვიდა მწყობრიდან.

⇨ მარშრუტის ოპტიმალობის მაჩვენებელი (inter-AS metric). ეს პარამეტრი საშუალებას იძლევა რომელიმე BGP მარშრუტიზატორმა შეატყობინოს რაიმე მარშრუტის სიგრძე მისი ავტონომიური სისტემის ფარგლებში. ეს ინფორმაცია გამოიყენება მოცემული ავტონომიური სისტემის მიმართ გარე მარშრუტიზატორების მიერ მარშრუტების მიზანშეწონილობის განსაზღვრისას.

- არსებობის შეტყობინება (keepalive message). ეს შეტყობინება არ შეიცავს დამატებით ველებს სტანდარტული შეტყობინების გარდა და გამოიყენება მარშრუტიზატორის მიერ, რათა პერიოდულად შეატყობინოს მეზობლებს საკუთარი მუშა მდგომარეობის შესახებ.
- შეცდომების შეტყობინება (error messages). ეს შეტყობინება გენერირდება, თუ აღმოჩენილია განსაკუთრებული სიტუაცია და მარშუტიზატორს სჭირდება შეატყობინოს თავის მეზობელს, თუ რატომ ხურავს ის მასთან შეერთებას. ეს შეტყობინება შეიცავს შეცდომების კოდებს, ქვეკოდებს და შეცდომების მონაცემებს.

ზოგადად სასაზღვრო მარშრუტიზატორში, რომელზეც მუშაობს BGP პროტოკოლი, სამარშრუტო ინფორმაციის მოძრაობის სტადიები ნაჩვენებია შემდეგ სქემაზე:



ყოველი მარშრუტიზატორი მიიღებს მარშრუტებს შიდა ან გარე მეზობლებისაგან. უმეტეს შემთხვევაში არ არის საჭირო და სასურველი, რომ მარშრუტების მთელი ნაკადი შევიდეს სამარშრუტო ტაბულაში. ამისათვის გამოიყენება შემომავალი ფილტრაციის მექანიზმი. ფილტრაცია შესაძლოა მოხდეს სახვადასხვა პარამეტრის მიხედვით: მაგალითად, IP პრეფიქსის ან გზის პარამეტრის მიხედვით. მარშრუტები გაფილტრვის შემდეგ გადაეცემა გადაწყვეტილების მიღების ბლოკს, რომელიც არკვევს ალტერნატიული მარშრუტებიდან თუ რომელი იხმაროს მოცემული დანიშნულების ადგილის მისაღწევად. იგი საზღვრავს ყოველი მარშრუტის სიგრძეს სხვადასხვა პარამეტრების მიხედვით და ადგენს ოპტიმალურ მარშრუტებს. გადაწყვეტილების მიღების პროცესს ადგილი აქვს BGP ტაბულაში, საიდანაც ოპტიმალური მარშრუტები შეიტანება სამარშრუტო ტაბულაში. სწორედ ეს მარშრუტები უშუალოდ გამოიყენება მონაცემთა მარშრუტიზაციის პროცესში და ისინი წარმოადგენენ კანდიდატებს გავრცელებაზე. ყოველთვის არ არის მიზანშეყონილი მთელი სამარშრუტო ტაბულის გავრცელება, რადგან ეს შეიძლება

იყონ ტრანზიტული ან მოცემული ავტონომიური სისტემის შიდა კერძო მარშრუტები. საუკეთესო მარშრუტები სამარშრუტო ტაბულიდან გადაეცემა გამავალ ფილტრს, რომელიც აარჩევს მეზობლებისათვის გადასაცემ მარშრუტებს, რის შემდეგაც ამუშავდება მარშღუტთა გავრცელების მექანიზმი.

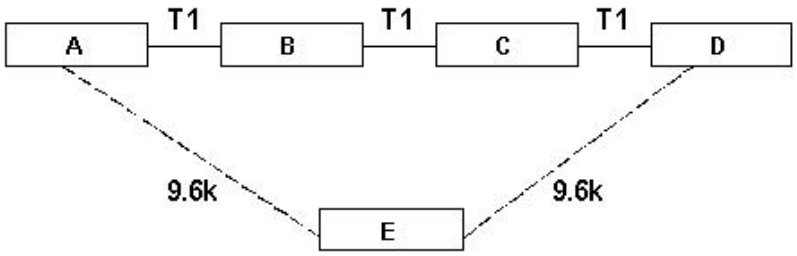
6. მარშრუტიზაციის პროტოკოლების შედარებითი ანალიზი

წინა თავში განხილული იყო მარშრუტიზაციის პროტოკოლთა მთელი ნუსხა, რომლებიც ამჟამად ფართოდ იხმარება TCP/IP ქსელებში პაკეტების მარშრუტიზაციისათვის. ყოველ მათგანს ახასიათებს თავისი გამოყენების სფერო, აქვს სპეციფიური შეზღუდვები და ახასიათებს დადებითი უარყოფითი მხარეები. ზემოთ განხილული მასალის საფუძველზე მოვახდინოთ ამ პროტკოლთა შედარებითი ანალიზი.

RIP პროტკოლი შეიქმნა იმ დროს, როცა ინტერნეტი ჯერ არ იყო ფართოდ გავრცელებული მსოფლიოში და ავტონომიური სისტემებით შემოიფარგლებოდა. სწორედ ამით აიხსნება მისი შედარებითი სიმარტივე და ის ნაკლოვანებები, რომლებიც მას გააჩნია. ის გამოიყენება პაკეტების მარშრუტიზაციისათვის ავტონომიური სისტემის შიგნით. ამჟამად ამ მიზნებისათვის უფრო მიზანშეწონილია OSPF პროტკოლის გამოყენება, რომელიც თუმცა უფრო რთულია, სამაგიეროდ ხასიათდება უკეთესი მოქნილობით და ქსელის რესურსების უკეთესი გამოყენებით.

RIP პროტოკოლის ძირითად ნაკლს წარმოადგენს მარშრუტის სიგრძის შეზღუდვა 16-ით, რაც მას გამოუსადეგარს ხდის დიდი ზომის ავტონომიურ სისტემებში, სადაც მარშრუტში ნახტომთა სიგრძე შესაძლოა აღემატებოდეს 15-ს. OSPF-ს ეს შეზღუდვა არ გააჩნია.

რადგანაც RIP დისტანციურ ვექტორული პროტოკოლია, იგი ხმარობს მარშრუტის სიგრძის განსაზღვრისათვის მხოლოდ ერთ კრიტერიუმს - მარშრუტში ნახტომთა რაოდენობას. ამის გამო RIP პროტკოლის მეშვეობით შეუძლებელია გათვალისწინებულ იქნას ქსელის ხაზების დატვირთულობა ან გამტარუნარიანობა. ამიტომ შესაძლებელია შემთხვევები, როცა პაკეტის მარშრუტიზაციისას მიღებული გადაწყვეტილება არ არის ოპტიმალური. განვიხილოთ შემდეგ სქემაზე მოყვანილი შემთხვევა:



ამ შემთხვევაში RIP პროტოკოლი ტრაფიკს A კვანძიდან D-საკენ გააგზავნის A-E-D მარშრუტით, რადგან იგი შეიცავს მხოლოდ 2 ნახტომს, იმ დროს, როცა A-B-C-D მარშრუტი შეიცავს 3 ნახტომს. იგი ვერ საზღვრავს, რომ არჩეული მარშრუტის გამტარუნარიანობა ნაკლებია სხვა მარშრუტის გამტარუნარიანობაზე და ამიტომ არჩეული მარშრუტი ვერ იქნება ოპტიმალური.

OSPF პროტკოლში მარშრუტის სიგრძის სიდიდე ეფუძნება ხაზების მდგომარეობის მახასიათებლებს, როგორიცაა ხაზის გამტარუნარიანობა, დაყოვნება და საიმედოობა. ამიტომ მისთვის A-B-C-D მარშრუტს უფრო ნაკლები სიგრზე ექნება, ვიდრე A-E-D გზას.

RIP პროტოკოლს სჭირდება საკმაო დრო სამარშრუტო ინფორმაციის გავრცელებისათვის შედარებით დიდ ქსელში, რადგან ის პერიოდულად აგზავნის

მთელ სამარშრუტო ტაბულას, რომელიც შეიძლება საკმაოდ დიდი ზომის იყოს. შესაბამისად დამატებით იტვირთება ქსელი. გარდა ამისა სამარშრუტო ცვლილებები ვრცელდება მთელს ავტონომიურ სისტემაში, რაც კიდევ უფრო ზრდის სამარშრუტო კონფლიქტების წარმოქმნის შესაძლებლობას.

OSPF-ის შემთხვევაში ვრცელდება არა მთლიანი ტაბულა, არამედ მხოლოდ კორექტირებები და შეტყობინებები არ სცდებიან ცალკეული უბნის ფარგლებს. რის გამოც ქსელის დატვირთულობა საკმაოდ მცირდება.

RIP-ს არ გააჩნია უშიშროების საკმარისი უზრუნველყოფა. ადვილი შესაძლებელია შემთხვევით ან წინასწარი განზრახვით არასწორი სამარშრუტო ინფორმაცია შეტანილ იქნას სამარშრუტო ტაბულაში. RIP-ს არ აქვს რაიმე მეზობლის იდენტიფიცირების საშუალება, გარდა მისი IP მისამართისა, რომელიც ადვილად შეიძლება შეცვლილ იქნას.

უშიშროების უზრუნველსაყოფად OSPF პროტოკოლში შემოღებულია 64 ბიტიანი აუთენტიკაციის ველი, რომლის საშუალებითაც ის ახდენს სამარშრუტო შეტყობინებების იდენტიფიცირებას და უცნობი კვანძებიდან მოსული ინფორმაციის უგულვებელყოფას.

გარდა ზემოთ ხსენებულისა OSPF-ის დიდი უპირატესობაა ის ფაქტი, რომ მას შეუძლია რამოდენიმე ტოლი სიგრძის მქონე მარშრუტის ერთდროული გამოყენება, რის საშუალებითაც უკეთესად ხდება ქსელის გამტარუნარიანობის გამოყენება. RIP პროტკოლს შეუძლია დროის ყოველ მომენტში აირჩიოს მხოლოდ ერთი მარშრუტი.

მიუხედავად ყოველივე ამისა, მაინც RIP რჩება პოპულარული თავისი სიმარტივისა და თავსებადობის გამო მცირე ზომის ქსელებში, მაგრამ ზემოთ აღნიშნული უპირატესობების წყალობით OSPF პროტოკოლი სულ უფრო და უფრო იკიდებს ფეხს ინტერნეტის მარშრუტიზაციის სისტემაში.

გარდა RIP და OSPF პროტოკოლებისა, წინა თავში განიხილებოდნენ გარე მარშრუტიზაციის პროტოკოლები EGP და BGP. როგორც ზემოთ აღინიშნა, პროტოკოლი შეიქმნა 1984 წელს და ხასიათდება სიმარტივით. მას არ შეუძლია დამოუკიდებელი გადაწყვეტილების მიღება მარშრუტების უპირატესობის შესახებ და ამიტომაც მას არ შეუძლია სამარშრუტო მარყუჟების აღმოჩენა და მოსპობა. იგი მხოლოდ ავრცელებს თავისი ავტონომიური სისტემის შესახებ ინფორმაციას სხვა ავტონომიურ სისტემებში. გარდა ამისა მას არ გააჩნია CIDR პროტოკოლის მხარდაჭერა, რაც საკმაოდ ზღუდავს მის გამოყენების არეს. კვანძებში ხდება მოსული ინფორმაციის დაგროვება და მარშრუტიზაციის პროცესი საკმაოდ ემგვანება სტატიკურს. მას სრულიად არ გააჩნია უშიშროების უზრუნველყოფის საშუალებები, ამიტომ სამარშრუტო ტაბულებში საკმაოდ ადვილია შეცდომითი ინფორმაციის შეყვანა. სამწუხაროდ, EGP პროტკოლი უკვე ვერ პასუხობს მარშრუტიზაციის ძირითად მოთხოვნებს და მისი გამოყენება ხდება ძალიან იშვიათ შემთხვევებში, როცა საჭიროა თავსებადობა უკვე არსებულ ქსელთან.

ამჟამად ინტერნეტში ფართო ხმარებაშია BGP პროტოკოლი. როგორც იყო ნაჩვენები, მისი სტრუქტურა შედარებით რთულია: მას გააჩნია აუთენტიკაციის ინფორმაციის გადაცემის ველები, რაც მას სანდოდ ხდის. სამარშრუტო მონაცრემთა გადაცემაში ეყრდნობა TCP პროტოკოლს და ხმარობს მისი უშიშროებისა და სანდო გადაცემის საშუალებებს. ეს მნიშვნელოვნად ამცირებს პროტკოლის სირთულეს. EGP-საგან განსხვავებით მას გააჩნია CIDR-ის მხარდაჭერა, ე.ი. კორექტულად მუშაობს ქსელურ ნიღბებთან. BGP-ს გააჩნია შემომავალი და გამავალი მარშრუტების ფილტრაციის საშუალება, რაც უზრუნველყოფს არასასურველი მარშრუტების აღმოჩენასა და წაკვეთას. BGP-ში

გათვალისწინებულია სამარშრუტო შეტყობინებებში გავლილი გზად ავტონომიური სისტემების შეტანა. ამ ინფორმაციის ანალიზის შედეგად ხდება სამარშრუტო მარყუჟების აღმოჩენა და მარშრუტიზაციის გასწორება.

როგორც დავინახეთ, BGP მნიშვნელოვნად სჯობია EGP პროტოკოლს და უმეტეს შემთხვევებში მიზანშეწონილია მისი გამოყენება ავტონომიურ სისტემებს შორის მარშრუტიზაციის ორგანიზებისას.

დასკვნა

მოცემულ ნაშრომში გაკეთებულია მცდელობა გამოკვლეულ იქნას, თუ როგორ მოქმედებს მარშრუტიზაცია ზოგადად ქსელებში და კერძოდ ქსელების ყველაზე ფართოდ გავრცელებულ ოჯახში - TCP/IP ქსელებში, როგორიცაა ინტერნეტი, შეჯამდეს და ურთიერთშედარებულ იქნას მათი დადებითი და უარყოფითი მხარეები, განხილულ იქნან მათ მუშაობასთან დაკავშირებით წამოჭრილი პრობლემები და დაისახონ ამ პრობლემათა გადაჭრის მეთოდები.

ნაშრომში განხილულია რამოდენიმე ყველაზე მნიშვნელოვანი ალგორითმი და ამ ალგორითმების ბაზაზე 4 ყველაზე ფართოდ ხმარებადი პროტოკოლი: ორი შიდა მარშრუტიზაციის - RIP და OSPF, ორი გარე მარშრუტიზაციის პროტოკოლი - EGP და BGP, განსაზღვრულია მათი გამოყენების სფეროები. დეტალურად არის გამოკვლეული ისეთი მოვლენა, როგორიცაა სამარშრუტო რხევები უმოკლესი გზის ალგორითმების გამოყენებისას, ნაჩვენებია მათი წარმოქმნის მიზეზები. მოყვანილია ალგორითმების საილუსტრაციო პროგრამული მაგალითები.

თუმცა საქართველოს ფარგლებში კომპიუტერული ქსელების მცირე სირთულის გამო ჯერ არ არის დანერგილი დინამიური მარშრუტიზაცია, უკვე დადგა დრო, როცა გაჩნდა საჭიროება მისი გამოყენებისა, რაც შეუძლებელია დინამიური მარშრუტიზაციის მოქმედების პრინციპების ღრმა ცოდნის გარეშე.

ლიტერატურა

1. B. Kagan. Computers, computer systems and networks. Mir Publishers Moscow, 1988
2. Dimitri Bertsekas, Robert Gallager. Data networks. Prentice Hall International Inc, 1987
3. В. Золотов. Протоколы Internet. BHV - Санкт-Петербург, 1998
4. Stan Schatt. Linking LANs. McGraw Hill, 1995
5. Bassam Halabi. Internet routing architectures. Cisco Press, 1997
6. Chris Lewis. Cisco TCP/IP Routing Professional Reference. McGraw Hill, 1997
7. Spohn, Darren L. Data network design. McGraw Hill, 1997

შინაარსი