

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

LABORATORIJ TELEKOMUNIKACIJA I INFORMATIKE 2
POUZDANOST TELEKOMUNIKACIJSKE MREŽE

Matej Jularić
Petra Pelajić
Karlo Pondeljak
Luka Šaler
Marina Šolčić
Teo Toplak

Zagreb, lipanj 2018.

Sadržaj

1. Opis zadatka.....	1
2. Implementacija alata i korisničko sučelje.....	3
2.1 Korištene tehnologije i alati.....	3
2.2. Korisničke upute.....	4

1. Opis zadatka

Cilj ove laboratorijske vježbe izrada je alata za proračun raspoloživosti/pouzdanosti mreže/sustava. Alat se treba sastojati od dva dijela: grafičkog sučelja u kojem se određuje izgled topologije mreže koju se želi evaluirati, te programskog koda koji u pozadini radi proračun raspoloživosti/pouzdanosti zadane topologije. Unutar alata potrebno je omogućiti dodavanje linkova i čvorova koji se mogu povezivati tako da tvore različite strukture. Svakom linku i čvoru u dodanoj topologiji moguće je definirati intenzitet kvarova λ (*failure rate*) te intenzitet popravaka μ (*repair rate*). Također, za proračun pouzdanosti potrebno je zadati period vremena t za koji se pouzdanost računa. Svakom linku potrebno je definirati njegovu duljinu.

Proračun raspoloživosti/pouzdanosti između pojedinih parova čvorova temelji se na metodi elementarnih putova. Putovi za pojedini par čvorova, koji se sastoje od skupova linkova i čvorova između navedena dva čvora mogu se odrediti na sljedeće načine:

- Potrebno je pronaći 2 puta, prvi najkraći i drugi najkraći, pri tome pazeći da je drugi neovisan od drugog (npr. Koristeći algoritam Dijkstre) - prvi put je „primarni“, a drugi put je „rezervni put“.
- Potrebno je proizvoljno definirati 2 puta, tj. skup linkova i čvorova koji tvore „primarni“ i „rezervni put“.
- Potrebno je pobrojiti sve putove za zadani par čvorova.

Proračun raspoloživosti/pouzdanosti se može vršiti na razini određenog para čvorova ili na razini cijele mreže.

- Ako se proračun vrši na razini para čvorova, korisnik može odabrati par čvorova za koji želi izračunati raspoloživost/pouzdanost te jednu od metoda za određivanje elementarnih puteva.
- Ako se proračun vrši na razini mreže, tada u proračun ulaze svi parovi čvorova koji mogu komunicirati te korisnik i ovdje može odrediti na temelju kojeg kriterija se određuju elementarni putevi.

Proračuni raspoloživosti mreže, prosječna raspoloživost (A_{av} – av -raspoloživost) i minimalna raspoloživost između svih parova čvorova ($A_{s,t}$ – s,t -raspoloživost) izračunavaju se iz raspoloživosti komunikacije između svih parova čvorova (vjerojatnost da je barem jedan put između para čvorova ispravan ili drukčije rečeno vjerojatnost disjunkcije putova između 2 čvora).

Da bi mogli izračunati raspoloživost komunikacije između dva čvora treba transformirati (raspregnuti) nedisjunktne putove iz disjunkcije u disjunktne putove odnosno članove koji u sebi

sadrže dijelove putova. Za raspredanje može se primijeniti algoritam Abrahama¹ ili neki drugi algoritam koji vrši raspredanje.

Na temelju ukupnog kapaciteta komunikacije između pojedinih čvorova te njene raspoloživosti, moguće je izračunati ukupni gubitak prometa u jednoj godini (npr. ako je raspoloživost neke komunikacije 0,99999, kapacitet prijenosa 1 Gbit/s, ukupno vrijeme u kojem usluga nije bila dostupna u godinu – srednje vrijeme ispada (MDT - *mean down time*) 5,26 min = 315,6 s. Prema tome, godišnji gubitak prometa je ukupni neostvareni promet navedene komunikacije 1 Gbit/s X 315,6 s X f = 189,36 Gbit, gdje je f faktor iskoristivosti kapaciteta (0,6). Svi rezultati za pojedinu topologiju se mogu spremati u izlazne datoteke. Isto tako, pojedine topologije se mogu spremati, ili učitavati one topologije koje su prije kreirane.

¹ Mikac, Branko ; Sever, Željka. "Telekomunikacije I Informatika 2. Tema: Pouzdanost komunikacijske mreže (RELNET)." (1995).

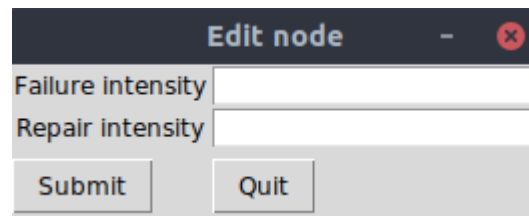
2. Implementacija alata i korisničko sučelje

2.1 Korištene tehnologije i alati

Odabrani programski jezik za ovaj zadatak je bio Python te je izgrađena *desktop* aplikacija. Uz pomoć programskih modula NetworkX, Pyglet i Fastnumbers napravili smo simulator.

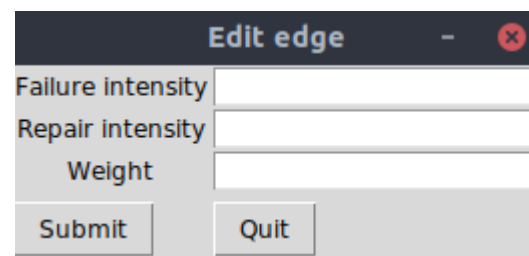
2.2. Korisničke upute

Unutar simulacijskog alata bitna činjenica je u kojem načinu rada se nalazite. Načini rada se lagano mijenjaju pritiskom tipke tipkovnice. Početni način rada omogućava dodavanje čvorova u mreži i njihovih parametara pritiskom miša na ekranu. Ekran dodavanja novog čvora prikazan je na Slici 1.



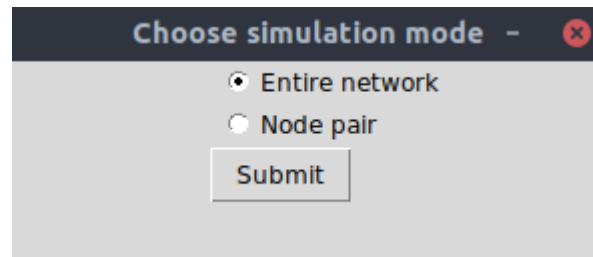
Slika 1. Dodavanje/Uređivanje novog čvora

Kako bi korisnik dodao brid između dva čvora potrebno je odabrati oba čvora te se prebaciti u način rada za uređivanje bridova. Bridovi također primaju svoje parametre koje je potrebno unijeti tijekom stvaranja novog brida. Ekran dodavanja novog brida je prikazan na Slici 2.



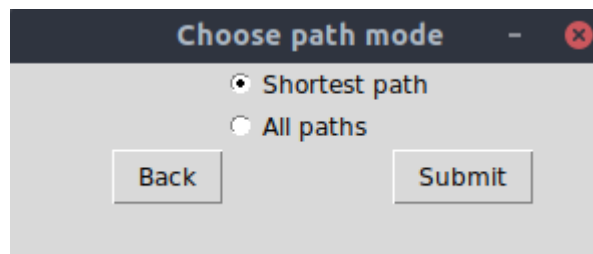
Slika 2. Dodavanje/Uređivanje novog brida

Prema tekstu zadatka korisnik može izabrati dva načina odrade simulacije. Na odabir mu je ponuđena opcija cijele mreže ili para čvorova. Na Slici 3. je prikazan ekran odabira.

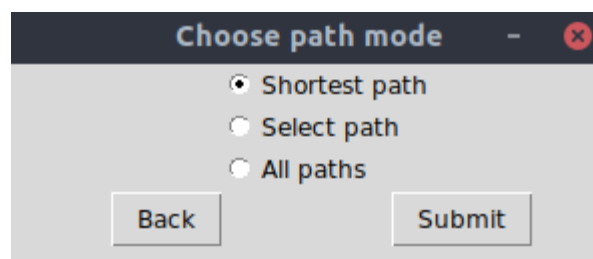


Slika 3. Način rada simulacije

Nakon što korisnik odabere način rada te u ovisnosti koji je način izabran prikazan mu je jedan od dva ekrana. U slučaju čitave mreže korisnik dobije na izbor opciju najkraćeg puta ili puteva za cijelu mrežu. Opcije za čitavu mrežu su prikazane na Slici 4. Ukoliko je izabrana opcija para čvorova prikazan je ekran na Slici 5. koji prikazuje opcije za najkraći put, odabir vlastitog puta te sve puteve.



Slika 4. Ekran opcija za čitavu mrežu



Slika 5. Ekran opcija za par čvorova

Nakon što je korisnik unio sve parametre elemenata simulacije te izabrao način odrade simulacije potrebno je samo unijeti vrijeme simulacije. Ekran za unos vremena simulacije je prikazan na Slici 6. Unosom vremena simulacije pojavljuje se ekran u kojem je korisniku dana opcija spremanja rezultata

simulacije na neku lokaciju na računalu. Odabirom lokacije spremanja se izvršava simulacija te sprema na izabranu lokaciju. Iskočni prozor također prikazuje rezultate simulacije te se rezultati također mogu naknadno vidjeti u običnoj tekstualnoj datoteci. Svi načini rada se mogu lagano vidjeti pritiskom tipke "h" nakon koje se prikazuje ekran s detaljnim opisom naredbi. Izbornik sa svim opcijama je prikazan na Slici 6.

Help

The following commands are available (just press the key for the command):

- c - press and hold in select_path mode to select start and end node
- d - switch to 'delete' mode (for deleting nodes and edges)
- e - switch to 'edge' mode (for adding edges)
- h - displays this help message
- i - switch to 'simulation' mode
- l - load graph, discarding current
- m - switch to 'modify' mode (for modifying attributes and move nodes)
- n - switch to 'node' mode (for adding nodes)
- p - hold p in select_path mode to select primary path
- q - quit program without saving
- r - hold r in select_path mode to select secondary path
- s - save graph
- t - input simulation time and start simulation
- y - redo last change
- z - undo last change

- esc - deselect current node (edge mode only)
- F11 - switch between full screen and windowed mode

There are also mouse commands:

- left click - do action according to current mode
- wheel scroll - increase reduce zoom
- mouse drag with right button pressed - move in space

Slika 6. Pomoćni izbornik

2.3 Analitičke funkcije

Funkcija `calculateReliability` prima graf i listu čvorova koja predstavlja puteve između čvorova. Ukupna pouzdanost linkova i čvorova računa se da se pomnože pouzdanosti svih čvorova i linkova. Na isti način se pomoću funkcije `calculateAvailability` računa raspoloživost komunikacije između zadanog para čvorova.

```
def calculateReliability(graph, path):
    reliability = 1
    for i in range(len(path)):
        reliability *= graph.node[path[i]]['R']

    for i in range(len(path) - 1):
        if i < len(path) - 1:
            reliability *= graph.get_edge_data(path[i], path[i + 1])['R']
    return reliability

def calculateAvailability(graph, path):
    availability = 1
    for i in range(len(path)):
        availability *= graph.node[path[i]]['A']

    for i in range(len(path) - 1):
        if i < len(path) - 1:
            availability *= graph.get_edge_data(path[i], path[i + 1])['A']
    return availability
```

Pomoću funkcije `dijkstraCalculation` se računa pouzdanost ili raspoloživost (ovisno o modu) puta između odabranog para čvorova tako da se pronađu 2 najkraća puta, primarni i sekundarni te zatim pomoću gore navedenih funkcija se računa pouzdanost ili raspoloživost između njih. Pomoću formule za paralelu putova računamo ukupnu pouzdanost, odnosno raspoloživost.

```
def dijkstraCalculation(graph, first, last, mode):
    min_path = nx.dijkstra_path(graph, first, last)
    H = graph.copy()
    for i in range(len(min_path)):
        if i < len(min_path) - 1:
            H.remove_edge(min_path[i], min_path[i + 1])

    not_min_path = nx.dijkstra_path(H, first, last)

    if mode == 'A':
        up = calculateAvailability(graph, min_path)
        down = calculateAvailability(graph, not_min_path)

    if mode == 'R':
        up = calculateReliability(graph, min_path)
        down = calculateReliability(graph, not_min_path)

    paralel_1 = up / (graph.node[min_path[0]][mode] * graph.node[min_path[len(min_path) - 1]][mode])
    paralel_2 = down / (graph.node[not_min_path[0]][mode] * graph.node[not_min_path[len(not_min_path) - 1]][mode])
    paralel_final = (1 - (1 - paralel_1) * (1 - paralel_2))

    final_a = graph.node[min_path[0]][mode] * graph.node[min_path[len(min_path) - 1]][mode] * paralel_final
    return final_a
```


Pomoću funkcije `averageAvailabilityDijkstra` koja prima parametar `graph` računa se prosječna raspoloživost među svim parovima čvorova u zadanom grafu.

```
def averageAvailabilityDijkstra(graph):
    nodes = graph.nodes()
    all_availabilities = []
    for node in nodes:
        for node1 in nodes:
            if node1 > node:
                all_availabilities.append(dijkstraCalculation(graph, node, node1, 'A'))
    sum_availabilities = 0
    for number in all_availabilities:
        sum_availabilities += number
    average = sum_availabilities / len(all_availabilities)
    return average
```

Na sličan način pomoću funkcije `stAvailabilityDijkstra` računamo minimalnu raspoloživost u grafu.

```
def stAvailabilityDijkstra(graph):
    nodes = graph.nodes()
    min_availability = 1000
    for node in nodes:
        for node1 in nodes:
            if node1 > node:
                current = dijkstraCalculation(graph, node, node1, 'A')
                if current < min_availability:
                    min_availability = current
    return min_availability
```

Pomoću funkcije `averageAvailabilityAbraham` koja prima parametar `graph` računamo raspoloživost svih puteva raspregnutih abrahamovim algoritmom između dva zadana čvora.

```
def averageAvailabilityAbraham(graph):
    nodes = graph.nodes()
    all_availabilities = []
    for node in nodes:
        for node1 in nodes:
            if node1 > node:
                paths = getAllPaths(graph, node, node1)
                all_availabilities.append(abraham(graph, paths, 'A'))
    sum_availabilities = 0
    for number in all_availabilities:
        sum_availabilities += number
    average = sum_availabilities / len(all_availabilities)
    return average
```

Na analogan način, pomoću funkcije `stAvailabilityAbraham` koja prima `graph` računamo minimalnu raspoloživost cijele mreže.

```
def stAvailabilityAbraham(graph):
    nodes = graph.nodes()
    min_availability = 1000
    for node in nodes:
        for node1 in nodes:
            if node1 > node:
                paths = getAllPaths(graph, node, node1)
                current = abraham(graph, paths, 'A')
                if current < min_availability:
                    min_availability = current
    return min_availability
```

Funkcija customPath koja prima parametre graph, path1, path2, mode koristi se u slučaju kada korisnik sam izabere puteve između čvorova. Pomoću nje računamo pouzdanost i raspoloživost istih.

```
def customPath(graph, path1, path2, mode):
    if mode == 'A':
        up = calculateAvailability(graph, path1)
        down = calculateAvailability(graph, path2)
    if mode == 'R':
        up = calculateReliability(graph, path1)
        down = calculateAvailability(graph, path2)

    paralel_1 = up / (graph.node[path1[0]][mode] * graph.node[path1[len(path1) - 1]][mode])
    paralel_2 = down / (graph.node[path2[0]][mode] * graph.node[path2[len(path2) - 1]][mode])
    paralel_final = (1 - (1 - paralel_1) * (1 - paralel_2))
    final = graph.node[path1[0]][mode] * graph.node[path1[len(path1) - 1]][mode] * paralel_final
    return final
```

Pomoću funkcije trasformationCalculate koja prima lamb, koja predstavlja intenzitet kvarova, mi koji predstavlja intenzitet popravaka te t koji predstavlja zadano vrijeme, računamo pouzdanost i raspoloživost iz zadanih parametara.

```
def transformationCalculate(lamb, mi, t):
    mttf = 1 / lamb
    mttr = 1 / mi
    a = mttf / (mttf + mttr)
    r = math.exp(-lamb * t)
    return a, r
```