

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

LABORATORIJ TELEKOMUNIKACIJA I INFORMATIKE 1

KOMUNIKACIJSKI PROTOKOLI

Ime Prezime

Zagreb, studeni 2011.

Sadržaj

1. Projektni zadatak	2
2. Specifikacija protokola u programskom jeziku Promela	3
3. Simulacija i verifikacija specificiranog protokola pomoću programskog alata Spin	7
4. Protokol modeliran Petrijevom mrežom	10
5. Analiza protokola programskim alatom DaNAMiCS	15
6. Usporedba rezultata dobivenih programskim alatima Promela/Spin i PetriNet/DaNAMiCS	16

1. Projektni zadatak

Specificirajte protokol za prijenos podataka tako da definirate tri vrste procesa: predajnik, međuspremnik i prijamnik koji međusobno komuniciraju sinkrono.

Predajnik i prijamnik izvedite tako da se u slučaju otkrivanja pogreške provodi retransmisija jedinice podataka. Prijamnik provodi provjeru ispravnosti svake primljene jedinice podataka te šalje potvrdu predajniku.

Međuspremnikom modelirajte medij s mogućom pogreškom u prijenosu jedinice podataka. Pretpostavite da ne dolazi do greške na potvrdi. Pokrenite dva predajnika i tri prijamnika. Svaki predajnik šalje po 10 poruka prijamnicima preko međuspremnika. Kada međuspremnik primi poruku od predajnika, odmah ju prosljeđuje prijamnicima. Međuspremnik naizmjenično prima poruke od predajnika, a po principu slučajnog odabira ih prosljeđuje prijamnicima.

U okviru projektnog zadatka potrebno je napraviti sljedeće:

- a) Specificirati protokol u programskom jeziku Promela.
- b) Specificirani protokol verificirati koristeći programski alat Spin i raspraviti rezultate.
- c) Modelirati zadani protokol pomoću Petrijeve mreže.
- d) Analizu protokola napraviti programskim alatom DaNAMiCS i raspraviti rezultate.
- e) Usporediti rezultate dobivene programskim alatima Promela/Spin i PetriNet/DaNAMiCS.

2. Specifikacija protokola u programskom jeziku Promela

U nastavku je dan programski kod u jeziku Promela koji specificira zadani protokol.

```
#define error -1;

init {                                // inicijalizacijski proces
    chan k1 = [0] of {int};
    chan k2 = [0] of {int};
    chan k3 = [0] of {int};
    chan k4 = [0] of {int};
    chan k5 = [0] of {int};
    run Sender(k1);                    // pokretanje predajnika, prijamnika i
    run Sender(k2);                    // međuspremnika s pripadnim kanalima
    run Buffer(k1,k2,k3,k4,k5);
    run Receiver(k3);
    run Receiver(k4);
    run Receiver(k5);
}

// predajnik
proctype Sender(chan k1){
    int count = 1; int confirm = 0;
    do
        :: count <= 10 ->
            k1!count; // predajnik šalje poruku međuspremniku
            k1?confirm; // predajnik prima potvrdu od međuspremnika
            if // provjera ispravnosti primljene potvrde
                :: confirm != count -> skip; // ako potvrda nije jednaka
                // poslanoj poruci, potrebna je retransmisija
                :: confirm == count -> count = count +1
            fi;
        :: else -> break
    od
}

// prijamnik
proctype Receiver(chan k2) {
    int count = 1;
```

```

int data;
int flagR = 1;
do
:: count <= 10 ->
    if
    :: flagR == 1 ->
        k2?data;    // prijamnik prima poruku od jednog
        k2!data;    // predajnika i šalje potvrdu
        if
        :: data != count -> skip;
        :: data == count -> flagR = 2
        fi;
    :: flagR == 2 ->
        k2?data;    // prijamnik prima poruku od drugog
        k2!data;    // predajnika i šalje potvrdu
        if
        :: data != count -> skip;
        :: data == count -> count = count +1; flagR = 1
        fi;
    fi;
:: else -> break
od
}
// međuspremnik
proctype Buffer(chan k1; chan k2; chan k3; chan k4; chan k5){
    int count = 1; int data = 0;
    int confirm3 = 0; int confirm4 = 0; int confirm5 = 0;
    int flag = 0; int state = 1;
    if
    :: flag = 1;    // odabir koji će predajnik prvi slati
    :: flag = 2
    fi;
do
:: count <= 10 ->
    if
    :: state == 1 -> // primanje poruke od prvog predajnika
        if
        :: flag == 1 -> k1?data;
        :: flag == 2 -> k2?data
        fi;

```

```

        if
// prosljeđivanje poruka prijateljima po principu slučajnog odabira
        :: k3!data;k4!data;k5!data;
        :: k3!data;k5!data;k4!data;
        :: k4!data;k3!data;k5!data;
        :: k4!data;k5!data;k3!data;
        :: k5!data;k3!data;k4!data;
        :: k5!data;k4!data;k3!data;
        :: k3!error;k4!error;k5!error; // generiranje pogreške
        :: k4!error;k3!error;k5!error;
        :: k5!error;k3!error;k4!error
        fi;
    if
// primanje potvrda od prijatelja po principu slučajnog odabira
        :: k3?confirm3;k4?confirm4;k5?confirm5;
        :: k3?confirm3;k5?confirm5;k4?confirm4;
        :: k4?confirm4;k3?confirm3;k5?confirm5;
        :: k4?confirm4;k5?confirm5;k3?confirm3;
        :: k5?confirm5;k3?confirm3;k4?confirm4;
        :: k5?confirm5;k4?confirm4;k3?confirm3
        fi;
    if //slanje potvrde predajniku koji je poslao poruku
        :: flag == 1 -> k1!confirm3;
        :: flag == 2 -> k2!confirm3
        fi;
    if //ako je potvrda ispravna, drugi predajnik može
        //poslati svoju poruku
        :: confirm3 != count -> skip;
        :: confirm3 == count -> state = 2
        fi;

```

Dalje analogno slijedi slanje poruke drugog predajnika, koji svoju poruku može poslati tek kada prvi predajnik primi pozitivnu potvrdu. Kada drugi predajnik dobije pozitivnu potvrdu, opet je prvi predajnik na redu za slanje. Zbog jednostavnosti, sve potvrde su iste, odnosno ili sve pozitivne ili sve negativne pa je predajniku dovoljno poslati samo jednu.

```

        :: state == 2 ->

        if
        :: flag == 1 -> k2?data;
        :: flag == 2 -> k1?data;
        fi;

```

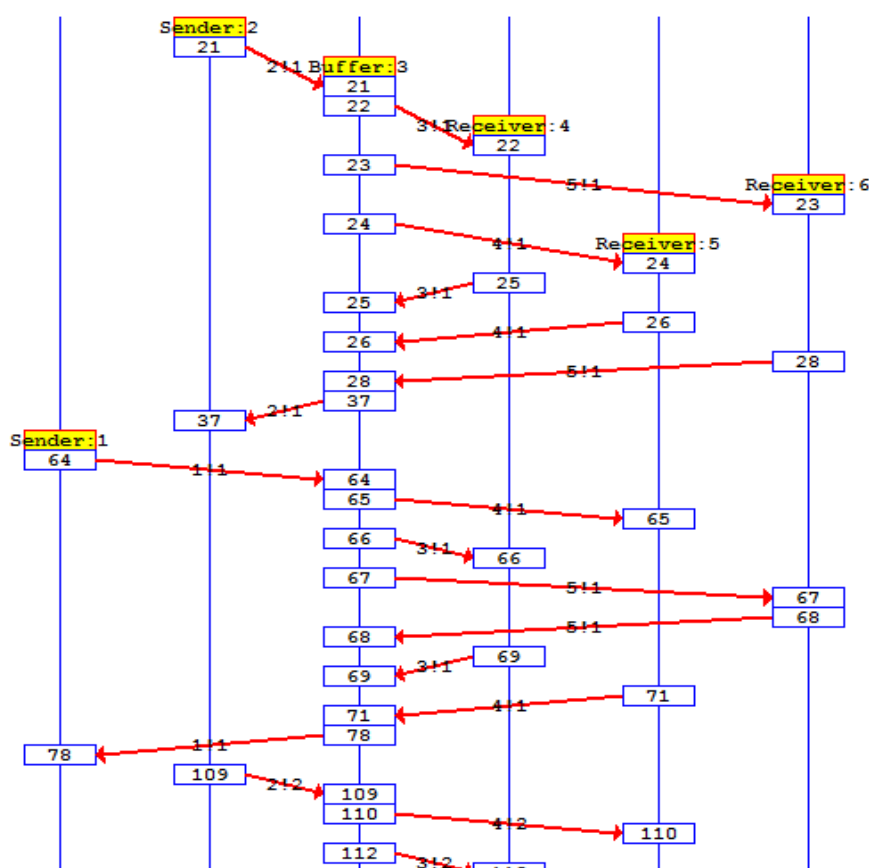
```

        if
        :: k3!data;k4!data;k5!data;
        :: k3!data;k5!data;k4!data;
        :: k4!data;k3!data;k5!data;
        :: k4!data;k5!data;k3!data;
        :: k5!data;k3!data;k4!data;
        :: k5!data;k4!data;k3!data;
        :: k3!error;k4!error;k5!error;
        :: k4!error;k3!error;k5!error;
        :: k5!error;k3!error;k4!error
        fi;
        if
        :: k3?confirm3;k4?confirm4;k5?confirm5;
        :: k3?confirm3;k5?confirm5;k4?confirm4;
        :: k4?confirm4;k3?confirm3;k5?confirm5;
        :: k4?confirm4;k5?confirm5;k3?confirm3;
        :: k5?confirm5;k3?confirm3;k4?confirm4;
        :: k5?confirm5;k4?confirm4;k3?confirm3
        fi;
        if
        :: flag == 1 -> k2!confirm3;
        :: flag == 2 -> k1!confirm3
        fi;
        if
        :: confirm3 != count -> skip;
        :: confirm3 == count -> state = 1; // ako je primljena
        count = count +1 // poz. potvrda, kreće nova iteracija
        fi;
    fi;
    :: else -> break // ako je svaki predajnik poslao svih 10 poruka
    od
}

```

3. Simulacija i verifikacija specificiranog protokola pomoću programskog alata Spin

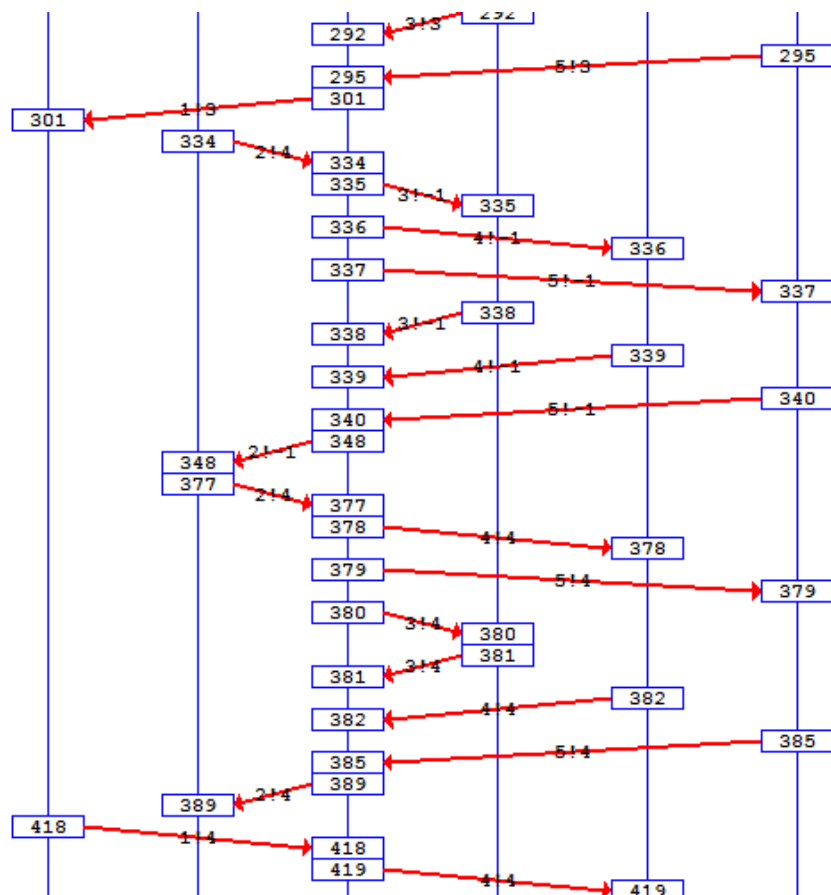
Na slici 3.1 prikazano je naizmjenično slanje poruka predajnika međuspremniku, prosljeđivanje tih poruka prijamnicima po principu slučajnog odabira i slanje pozitivnih potvrda predajnicima. Najprije 2. predajnik šalje poruku međuspremniku koji ju odmah prosljeđuje 1., 3. pa 2. prijamniku, zatim oni šalju potvrde međuspremniku, koji potvrdu šalje predajniku. Nakon što je 2. predajnik primio ispravnu potvrdu, 1. predajnik može poslati svoju poruku međuspremniku, koji šalje poruku 2., 1. pa 3. prijamniku.



Slika 3.1 Predajnici naizmjенично šalju poruke међuspremniku koji ih
prosljeђује prijammnicima po principu slučajnog odabira

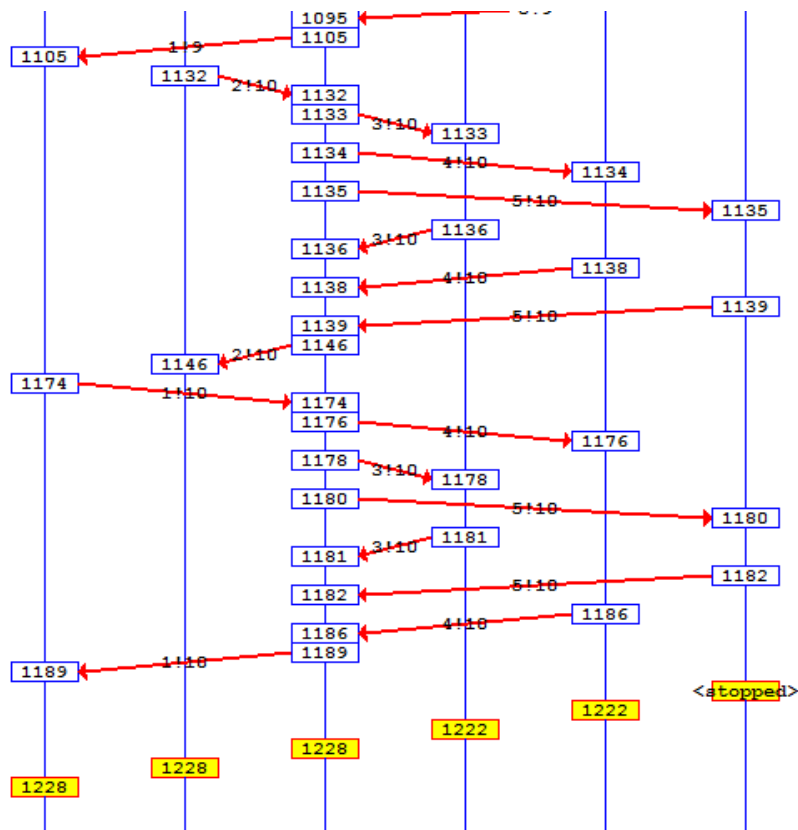
U slučaju da međuspremnik pošalje neispravnu poruku prijateljima, predajniku se šalje negativna potvrda i slijedi ponovno slanje iste poruke, odnosno retransmisija, kao što je prikazano na slici 3.2. Predajnik za poruku 4 dobije odgovor -1 pa opet međuspremniku

šalje poruku 4. Nakon što je primio pozitivnu potvrdu, koja je identična poslanoj poruci, drugi predajnik šalje svoju poruku.



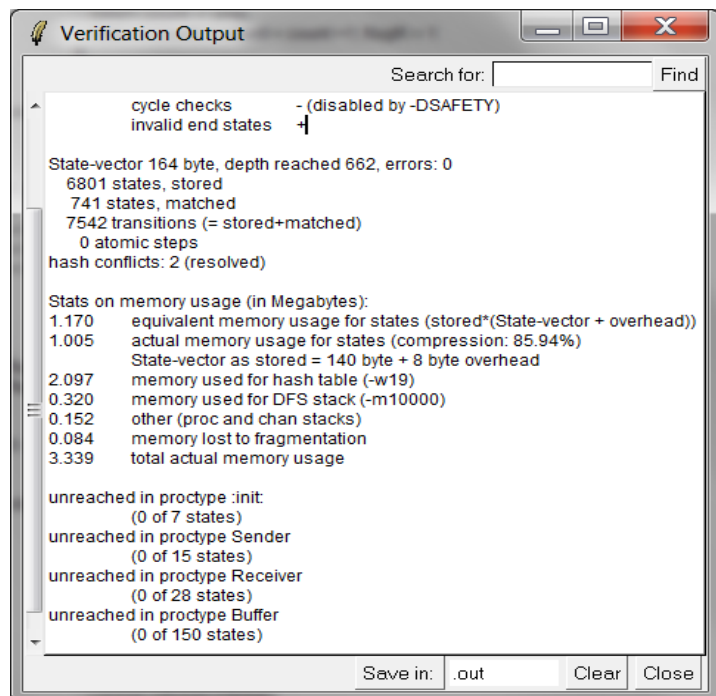
Slika 3.2 Slučaj retransmisije

Slika 3.3 prikazuje kraj procesa. Oba su predajnika ispravno poslala svih 10 poruka, prijammnici su ih primili i poslali potvrde te se svi procesi zaustavljaju.



Slika 3.3 Završetak procesa

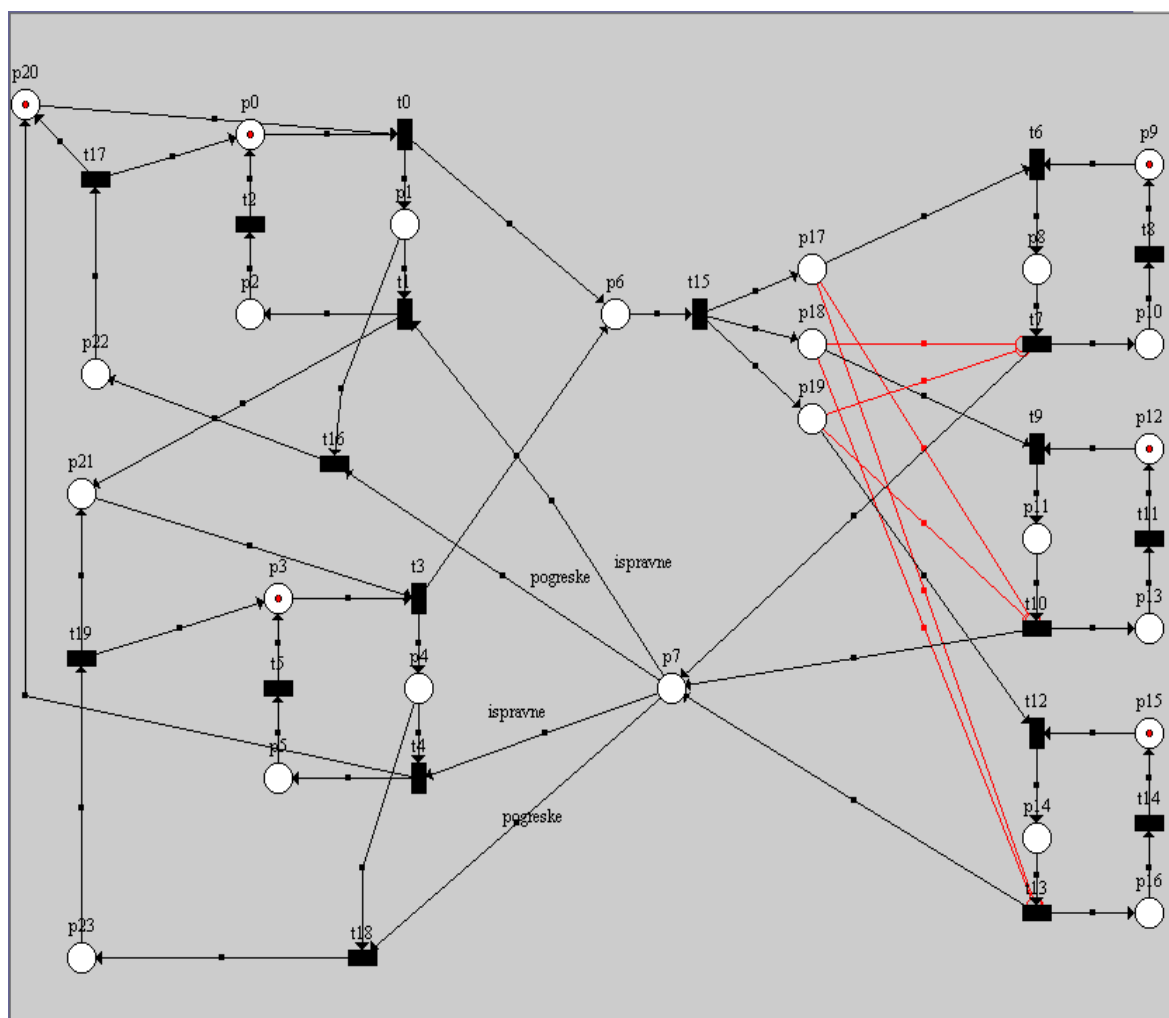
Verifikacija protokola pomoću programskog alata Spin prikazana je na slici 3.4. Pokazuje da u specificiranog protokolu nema grešaka niti nedohvatljivih stanja ni u jednom procesu.



Slika 3.4 Rezultat verifikacije protokola

4. Protokol modeliran Petrijevom mrežom

Slika 4.1 prikazuje Petrijevu mrežu kojom je modeliran zadani protokol. Na lijevoj se strani nalaze 2 predajnika, na desnoj 3 prijamnika, a između njih se nalazi međuspremnik. Dodatno, predajnici imaju kontrolna mjesta koja određuju koji je predajnik na redu za slanje poruke međuspremniku. Stigne li predajniku negativna potvrda, opet će on slati poruku, odnosno doći će do retransmisije, a stigne li mu pozitivna potvrda, slanje poruke bit će omogućeno drugom predajniku.



Slika 4.1 Petrijeva mreža

U nastavku su opisana mjesta i prijelazi modelirane mreže.

Mjesta

p0, p3	predajnici su spremni za slanje poruka međuspremniku
p9, p12, p15	prijamnici su spremni za primanje poruka od međuspremnika
p1, p4	predajnik je poslao poruku
p6	poruka u međuspremniku (na kanalu)
p17, p18, p19	po jedna poruka spremna za slanje na kanal svakog prijamnika
p8, p11, p14	prijamnik je primio poruku
p10, p13, p16	prijamnik je poslao potvrdu
p7	potvrde u međuspremniku
p2, p5	predajnik je primio pozitivnu potvrdu
p23, p22	predajnik je primio negativnu potvrdu
p19, p20	kontrolna mjesta koja određuju koji će predajnik slati poruku

Prijelazi

- t0, t3 – predajnik šalje poruku međuspremniku

p20, p21	p0, p3	p6	p1,p4
●	●	●	● ●

- t15 – međuspremnik šalje poruke prijateljima

p6	p17	p18	p19	slanje:
●	●	●	●	ispravne poruke
●	●	●	●	pogrešne poruke

- t6, t9, t12 – predajnik prima poruku ili pogrešku od međuspremnik

p9, p12, p15	p17, p18, p19	p8, p11, p14	primitak poruke:
●	●	●	ispravna poruka
●	●	●	pogrešna poruka

- t7, t10, t13 – prijatelj šalje pozitivnu ili negativnu potvrdu međuspremniku

p17,p19	p18, p17	p14, p11, p8	p7	p16,p13, p10	potvrda:
- ●	- ●	●	●	●	pozitivna
- ●	- ●	●	●	●	negativna

- t8, t11, t14 – unutarnji prijelazi prijatelja

p10, p13, p16	p9, p12, p15	prijatelj primio:
●	●	ispravnu poruku
●	●	pogrešnu poruku

- t2, t5 – unutarnji prijelazi predajnika nakon što primi pozitivnu potvrdu

p2, p5	p0, p3
●	●




- t1, t4 – predajnik prima pozitivnu potvrdu

p7	p1, p4	p20, p21	p2, p5
● ● ●	● ●	●	●

- t16, t18 – predajnik prima negativnu potvrdu, slijedi retransmisija

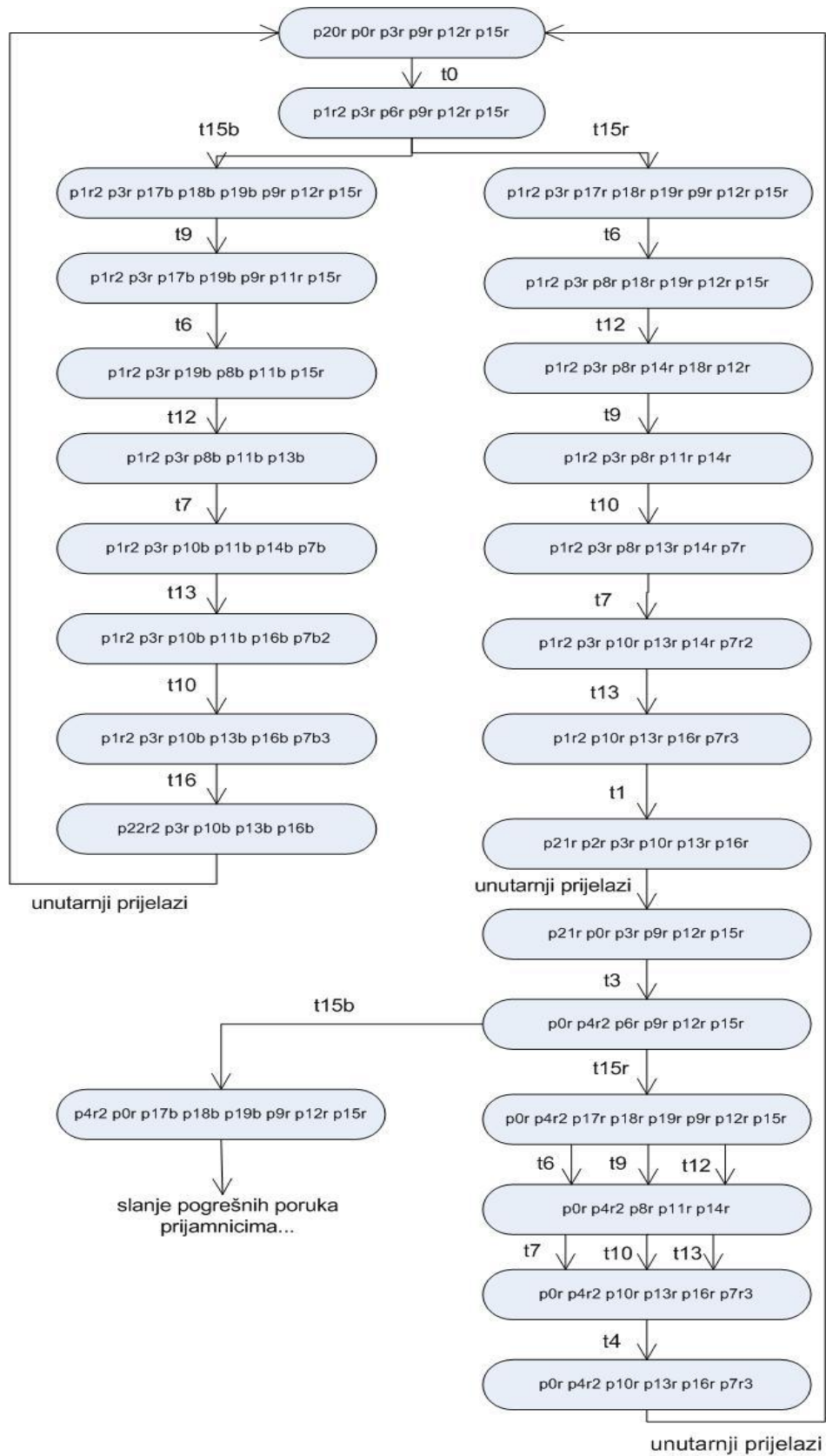
p1, p4	p7	p22, p23
● ●	● ● ●	● ●

- t17, t19 – omogućava retransmisiju predajniku koji je primio negativnu potvrdu

p22, p23	p0, p3	p20, p21
		

Graf stanja

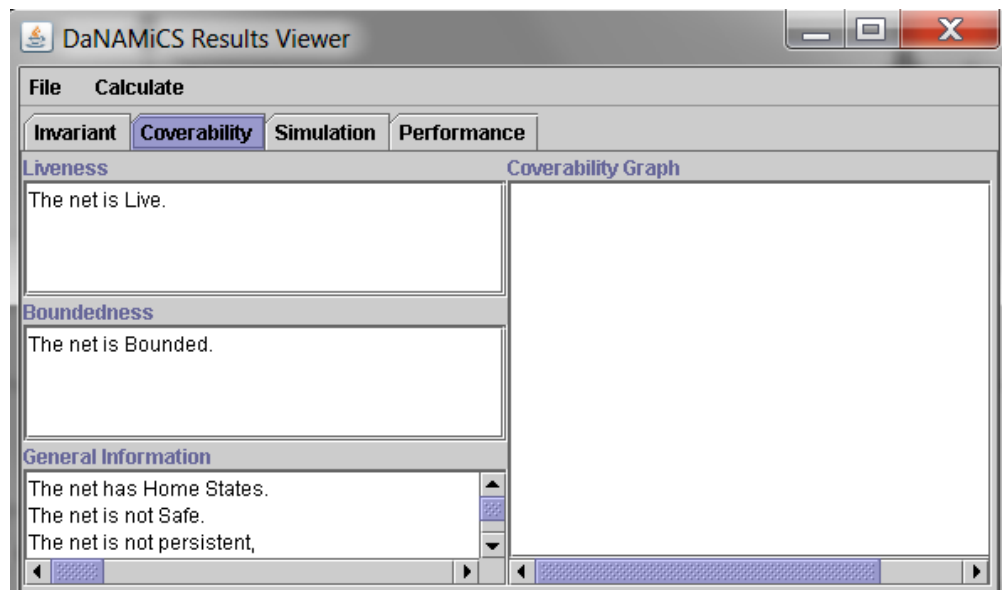
Slika 4.2 prikazuje graf stanja Petrijeve mreže. Iza oznake mjesta slijedi boja i broj obojenih oznaka, npr. p20r označava mjesto p20 koje ima jednu crvenu oznaku, a p18b mjesto koje ima jednu plavu oznaku. Na početku oznaka u kontrolnom stanju p20 omogućava prvom predajniku slanje poruka. Međuspremnik primi poruku te može poslati ispravnu poruku ili generirati pogrešku pa prijateljima poslati pogrešnu poruku. Prijelaz t15b označava slanje pogreške. Nakon što svi prijatelji prime poruku, vraćaju negativnu potvrdu međuspremniku, koji je prosljeđuje prvom predajniku. Prvi predajnik tada opet šalje istu poruku. Kada međuspremnik pošalje ispravnu poruku prijateljima, odnosno kada se dogodi prijelaz p15r, od njih dobije pozitivne potvrde i vraća prvom predajniku potvrdu te postavlja oznaku u kontrolno stanje p21, koje drugom predajniku omogućava slanje poruke. Kada drugi predajnik pošalje poruku, analogno prethodno opisanom slučaju, međuspremnik može generirati pogrešku, što će rezultirati retransmisijom poruke, ili poslati ispravnu poruku pa će idući na redu za slanje biti prvi predajnik. Na taj način predajnici naizmjenično šalju poruke međuspremniku. Slanje pogrešne poruke drugog predajnika na grafu nije do kraja prikazano jer je gotovo isto kao i za prvi predajnik. Budući da međuspremnik prijateljima šalje poruke po principu slučajnog odabira, postoji 6 različitih kombinacija, no zbog veličine grafa stanja, prikazana je samo jedna. Također, stanja u koja se prelazi unutarnjim prijelazima nisu prikazana jer bi to dodatno zakompliciralo i povećalo graf.



Slika 4.2 Graf stanja Petrijeve mreže

5. Analiza protokola programskim alatom DaNAMiCS

Slika 5.1 prikazuje rezultat simulacije provedene alatom DaNAMiCS.



Slika 5.1 Rezultat simulacije

Svojstva mreže:

- mreža je aktivna – ne postoje prijelazi koji se nikad ne izvode niti mjesta iz kojih se ne može izvesti niti jedan prijelaz,
- mreža nije sigurna – broj oznaka u nekim mjestima je veći od 1, primjerice u mjestima p1, p4 i p7,
- mreža je 3-ograničena – u mjestu p7 nalaze se 3 oznake koje označavaju potvrde prijavnika.
- mreža je reverzibilna – iz svakog stanja može se vratiti u početno stanje i
- mreža nije perzistentna – u stanju [p1r2 p3r p6r p9r p12r p15r] izvedbom prijelaza t15r, koje označava slanje ispravnih poruka prijateljima, onemogućava se izvedba prijelaza t15b, koje označava slanje pogrešnih poruka.

6. Usporedba rezultata dobivenih programskim alatima Promela/Spin i PetriNet/DaNAMiCS

Pomoću oba alata moguće je precizno specificirati zadani protokol, no postoje neke razlike. Kod simulacije protokola u alatu Spin, odnosno izvršavanja programskog koda napisanog u programskom jeziku Promela, moguće je vidjeti slijedni dijagram koji točno prikazuje redoslijed slanja i primanja poruka i potvrdi. Unutar prozora „*Simulation output*“ prikazani su koraci izvođenja simulacije.

Nakon crtanja Petrijeve mreže koja modelira protokol u alatu DaNAMiCS, korisnik pokreće animaciju u kojoj klikom miša može kontrolirati izvođenje i vidjeti što se događa u svakom koraku. Također, ako postoji više prijelaza koji se mogu izvesti, korisnik može sam odabrati redoslijed klikom na određeni prijelaz, dok se u alatu Spin sve odvija automatski, ovisno o tome kako je definirano u programskom kodu. Primjerice, ako međuspremnik šalje poruke prijateljima po principu slučajnog odabira, u animaciji u alatu DaNAMiCS korisnik svaki put sam odabire redoslijed, dok kod simulacije protokola u Spinu korisnik nema utjecaja na izvođenje prijelaza, nego su vjerojatnosti određene u programskom kodu. Isto vrijedi i za generiranje pogrešnih poruka.

Prilikom animacije u alatu DaNAMiCS moguće je neograničeno slati poruke, odnosno izvesti animaciju proizvoljan broj puta, dok se u alatu Spin pošalje onoliko poruka koliko je definirano u kodu, nakon čega simulacija završava. Programski alat Spin omogućuje verifikaciju specificiranog protokola u kojoj se može vidjeti postoje li greške i nedohvatljiva stanja. Simulacija modelirane Petrijeve mreže ispisuje njena svojstva, poput ograničenosti, sigurnosti, aktivnosti i perzistentnosti te ispiše graf stanja.