SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

LABORATORIJ TELEKOMUNIKACIJA I INFORMATIKE 1

KOMUNIKACIJSKI PROTOKOLI

Izvještaj projektnog zadatka

Zagreb, studeni, 2015.

Sadržaj

1. Projektni zadatak.....1
2. Model protokola u programskom jeziku Promela 2
3. Model protokola pomoću Petrijeve mreže u alatu CPN Tools 11
4. Usporedba.....22

1. Projektni zadatak

Modelirati protokol za prijenos podataka između predajnika i prijamnika preko komunikacijske mreže prema specifikaciji u nastavku.

SPECIFIKACIJA PROTOKOLA

Predajnik šalje pakete prijamniku preko komunikacijske mreže. Prijamnik provodi provjeru rednog broja svakog primljenog paketa te šalje potvrdu predajniku. Komunikacija između predajnika i prijamnika je sinkrona, predajnik šalje novi paket tek kad primi potvrdu o primitku prethodnog paketa. Predajnik izvedite tako da se u slučaju isteka određenog vremena predviđenog za primanje potvrde provodi retransmisija paketa.

Komunikacijsku mrežu modelirajte tako da postoji mogućnost gubitka u prijenosu jedinice podataka. Moguć je gubitak i paketa i potvrde.

Pokrenite dva predajnika i dva prijamnika. Predajnik 1 šalje pakete prijamniku 2. Predajnik 2 šalje pakete prijamniku 1. Vjerojatnost za gubitak paketa je 0.25, a za gubitak potvrde 0.33. Komunikacijska mreža prosljeđuje pakete naizmjenično. Kada proslijedi paket od jednog predajnika, sljedeći paket obavezno prosljeđuje od drugog predajnika sve dok jedan predajnik ne dovrši slanje. Potvrde se prosljeđuju slučajnim odabirom.

U Promeli specificirajte poruku u obliku 15 paketa bez sadržaja. U alatu CPN Tools definirajte da predajnik 1 šalje poruku „Dubrovnik“, a predajnik 2 poruku „Makarska“. Svaki paket sadrži maksimalno 1 znak iz poruke.

2. Model protokola u programskom jeziku Promela

Slijedi kod u programskom jeziku Promela.

```
mtype = {packet, reply};

init {
chan ch1send = [0] of {int, int, mtype, int};
chan ch2send = [0] of {int, int, mtype, int};
chan ch1recv = [0] of {int, int, mtype, int};
chan ch2recv = [0] of {int, int, mtype, int};

run Sender (1,2,ch1send);
run Sender (2,1,ch2send);
run Network(ch1send, ch2send, ch1recv, ch2recv);
run Receiver(ch1recv);
run Receiver(ch2recv);
}

proctype Sender (int sender; int receiver; chan ch) {
int count = 0;
int confirm;
mtype porr;

do //dok nisu svi paketi poslani radi sljedeće
:: (count < 15) ->
        porr = packet;
        //stavi na kanal
        ch!sender,receiver,porr,count
        if
        :: timeout -> ch!sender,receiver,porr,count //retransmisija
        :: ch?sender,receiver,porr,confirm; count = confirm //zablokiran čeka potvrdu
        fi
:: else -> break
od
}
```

```
proctype Network(chan ch1send; chan ch2send; chan ch1recv; chan ch2recv) {
int sender;
int receiver;
int data;
int who;
chan out;
int count1;
int count2;
mtype porr;

//odredi koji sender šalje prvi
if
::who = 1
::who = 2
fi;

do
::if
	:: (who == 1 && count2 < 15) -> //ako je na redu prvi sender za slanje, a drugi još nije poslao sve pakete
		if
		:: ch1send?sender,receiver,porr,data; out = ch2recv; who = 2
		//pročitaj što je prvi sender stavio na kanal, izlazni kanal postavi na ch2recv i stavi red na drugog
		:: ch1recv?sender,receiver,porr,data; out = ch2send; count2 = data
		:: ch2recv?sender,receiver,porr,data; out = ch1send; count1 = data
		fi

	:: (who == 1 && count2 == 15) ->
		if
		:: ch1send?sender,receiver,porr,data; out = ch2recv;
		:: ch2recv?sender,receiver,porr,data; out = ch1send; count1 = data
		fi

	:: (who == 2 && count1 < 15) ->
		if
		:: ch2send?sender,receiver,porr,data; out = ch1recv; who = 1
		:: ch1recv?sender,receiver,porr,data; out = ch2send; count2 = data
		:: ch2recv?sender,receiver,porr,data; out = ch1send; count1 = data
		fi
```

```
            :: (who == 2 && count1 == 15) ->
                        if
                        :: ch2send?sender,receiver,porr,data; out = ch1recv;
                        :: ch1recv?sender,receiver,porr,data; out = ch2send; count2 = data
                        fi
fi;

if
//ako je prvi sender dovršio slanje, a nije postavljen red na drugog, postavi ga
:: (count1 == 15 && who != 2) -> who = 2
:: (count2 == 15 && who != 1) -> who = 1
:: else-> skip
fi

if
:: (porr == packet) -> //vjerojatnost gubitka poruke = 0.25, zadnja se ne gubi
            if
            :: out!sender,receiver,porr,data
            :: out!sender,receiver,porr,data
            :: out!sender,receiver,porr,data
            :: (data!=14) -> skip
            fi
:: (porr == reply) -> //vjerojatnost gubitka potvrde = 0.33, zadnja se ne gubi
            if
            :: out!sender,receiver,porr,data
            :: out!sender,receiver,porr,data
            ::(data!=15) -> skip
            fi
fi;

if
:: (count1 == 15 && count2 == 15) -> break
:: else -> skip
fi

od

}
```

```
proctype Receiver (chan ch) {
int sender;
int receiver;
int data = 0;
mtype porr;

do
:: (data < 15) ->
        ch? sender,receiver,porr,data
        porr = reply;
        data = data + 1
        ch! sender,receiver,porr,data
:: else -> break
od
}
```

Kod pokretanja modela uvijek se prvo izvršava proces *init*. Definiraju se kanali *ch1send* (kanal između predajnika 1 i mreže), *ch2send* (kanal između predajnika 2 i mreže), *ch1recv* (kanal između prijamnika 1 i mreže) te kanal *ch2recv* (kanal između prijamnika 2 i mreže). Pokreće se 5 procesa, jedan za svakog predajnika, jedan za svakog prijamnika i jedan za mrežu. Kanali su prazni tako da se omogući sinkronizacija, odnosno ni jedan predajnik ni prijamnik ne stavljaju zapravo poruke na kanal već kad to pokušaju ostaju zablokirani dok neki drugi proces ne pročita što su pokušali staviti na kanal. Isto je i s čitanjem s kanala, kad proces pokuša čitati s kanala ostaje zablokiran dok neki drugi proces ne pokuša staviti poruku na kanal. Kad se to dogodi, proces koji je pokušao čitati s kanala pročita poruku koju je drugi pokušao staviti i oba se odblokiraju. U procesu koji modelira predajnike definiraju se varijable *count* i *confirm*. *count* je broj poruke koja se šalje (ujedno je to i poruka), a počinje od 0 i ide do 14, a *confirm* broj potvrde koju predajnici primaju. *confirm* je u rasponu od 1 do 15 i za jedan je veći od broja poruke za koju potvrđuje da je uspješno primljena (za poruku 0 potvrda je 1).

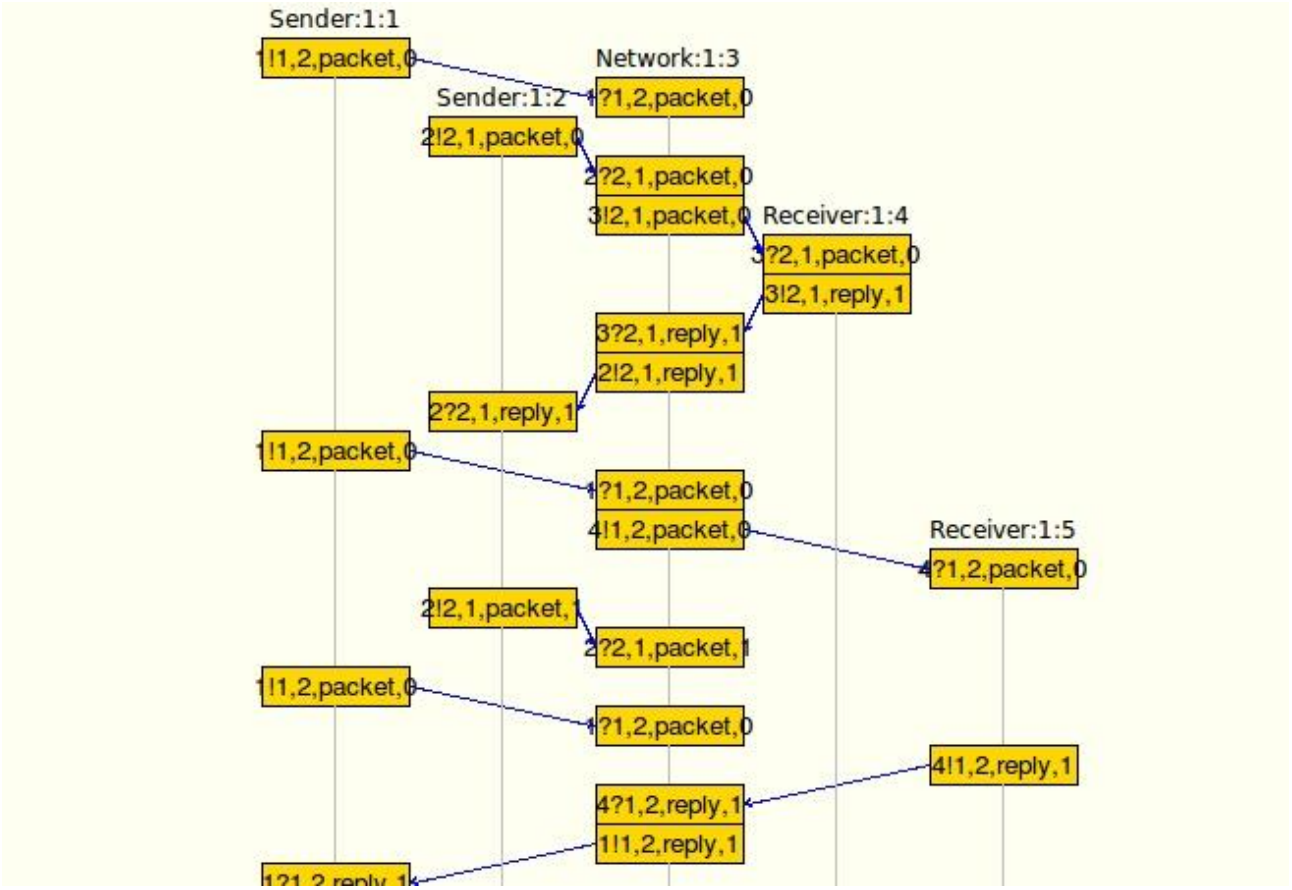
Svaki predajnik na kanal (*ch1send* ili *ch2send*) stavlja ID predajnika (*sender* – 1 ili 2), ID prijamnika (*recv* – 1 ili 2), tip poruke *porr* (*packet* ili *reply*; paket ili potvrda) te

varijablu *count*. Ako istekne vrijeme prije nego predajnik primi potvrdu, obavlja se retransmisija (*timeout*), odnosno predajnik ponovno stavlja poruku na kanal.

U mreži se prvo slučajno odlučuje od čijeg će predajnika prvog mreža proslijediti poruku, što označava varijabla *who* koja pohranjuje ID predajnika. Zatim se provjerava koji predajnik je na redu za slanje (jer moraju naizmjenično slati poruke). Npr. ako je na redu predajnik 1 (*who* == 1) i predajnik 2 još nije dovršio slanje (*count2* < 15), tada se s kanala *ch1send* pročita poruka koju je „postavio" predajnik 1, kao izlazni kanal se postavi *ch2recv* (jer predajnik 1 šalje pakete prijamniku 2, a prijamnik 2 ih prima po kanalu c*h2recv*) i prebaci se red na predajnika 2 (*who* = 2). Također se mogu čitati potvrde (ako su poslane) s kanala *ch1recv* i *ch2recv* koje prijamnici šalju slučajnim redoslijedom. Ako je na redu predajnik 1, a predajnik 2 je dovršio slanje (*count2* == 15), tada se samo čita poruka koju je postavio predajnik 1 i moguća pristigla potvrda s kanala *ch2recv*. Isto se događa i kod obratnog slučaja, kada je na redu za slanje predajnik 2. Zatim se provjeravaju slučajevi je li koji predajnik dovršio slanje, a nije prebacio red na onog drugog – tada dodjeljujemo pravo slanja onom predajniku koji nije dovršio slanje. Ako šaljemo paket (a ne potvrdu), vjerojatnost gubitka mora biti 0.25 i to se osigurava sa 3 naredbe (uspješnog) postavljanja na izlazni kanal i jednom „beznačajnom" koja označava gubitak paketa. Ta naredba je jedna od četiri, a ¼ = 0.25 i to je upravo vjerojatnost gubitka, osim u slučaju kada stigne posljednji paket – tada pretpostavljamo da se on neće izgubiti. Slično je i s gubitkom potvrde, ali je tad vjerojatnost 0.33 pa imamo dvije naredbe ispravnog slanja i jednu koja označava gubitak (osim u slučaju kada je to zadnja potvrda). Kada oba predajnika završe slanje, proces mreže se gasi.

U procesu koji modelira prijamnike događa se sljedeće: ako poruka koju je prethodno primio nije posljednja, čita s kanala preko kojeg komunicira s mrežom kako bi pročitao nadolazeće poruke. Podatak (*data*) koji primi uveća za 1 i to predstavlja potvrdu. Zatim na kanal postavi ID predajnika koji mu je poslao poruku, svoj ID, tip poruke (*reply* za potvrdu) te vrijednost potvrde, odnosno broj iduće poruke koju treba primiti.

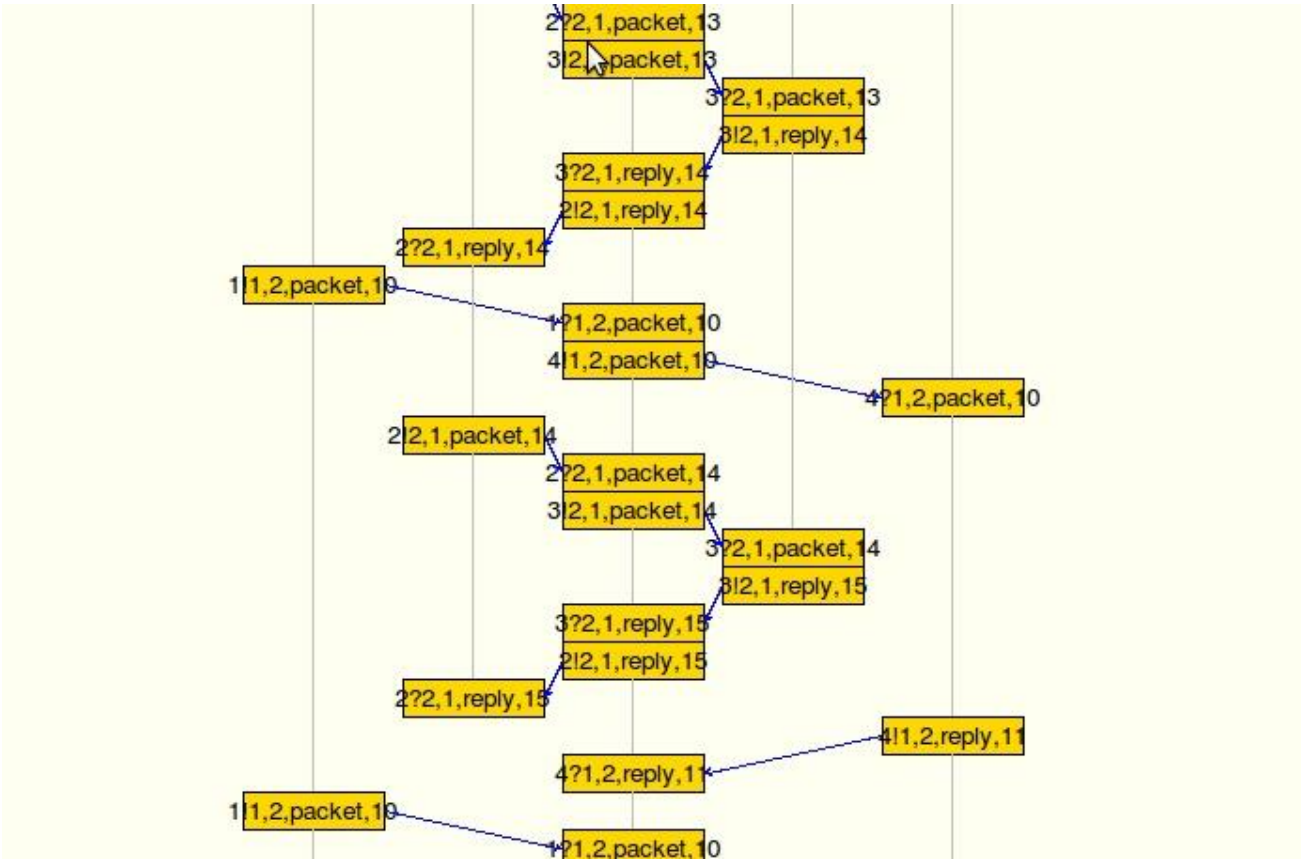
U nastavku slijede slike simulacije.



Slika 1

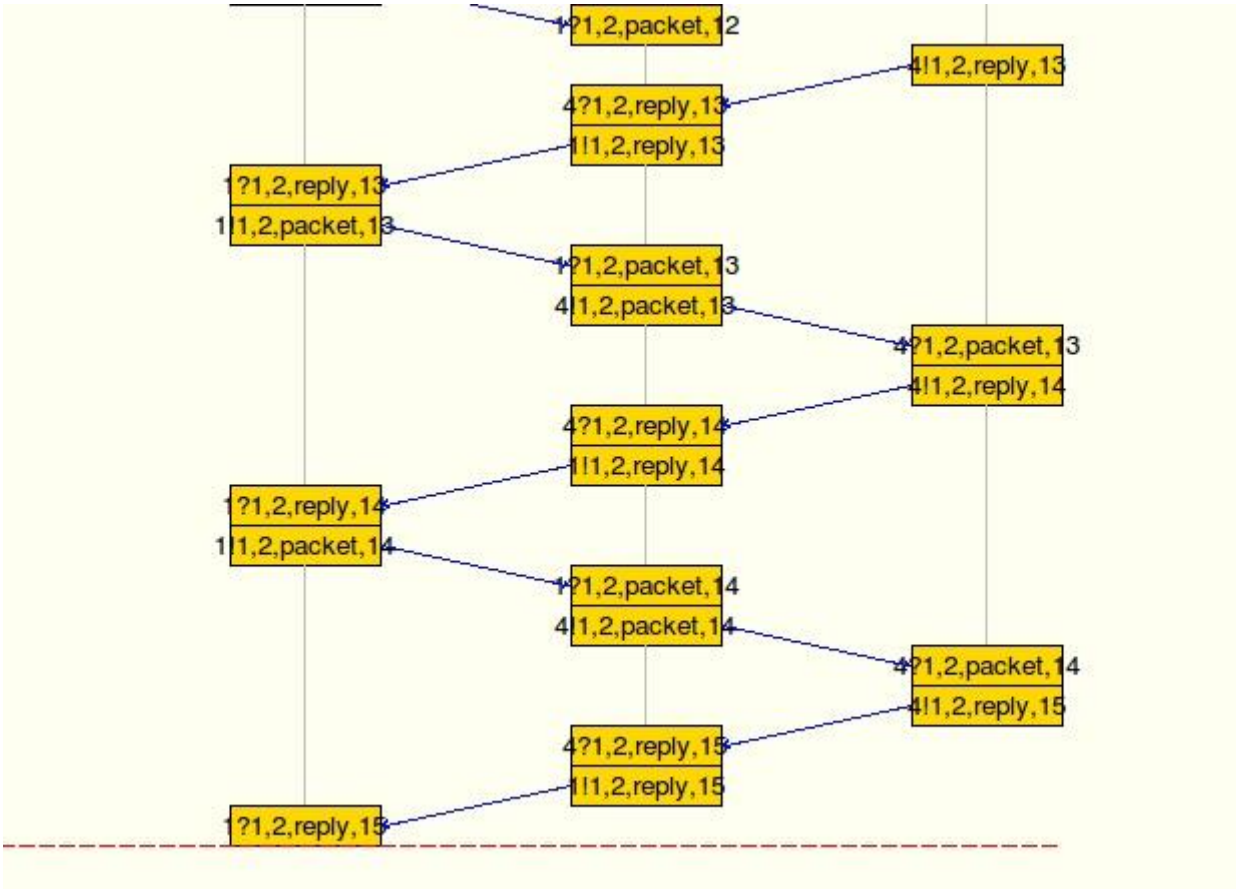
Na slici *Slika 1* prikazan je početak komunikacije. Vidljivo je da predajnici naizmjenično šalju poruke, a potvrde se prosljeđuju slučajnim redoslijedom. Vidljivi su i gubitci paketa, npr. gubitak paketa s rednim brojem 0 od predajnika 1 namijenjenog prijamniku 2.

Slika 2 prikazuje završetak slanja pak od predajnika 2, nakon čega više nema naizmjeničnog slanja nego se dalje prosljeđuju svi paketi predajnika 1. Vidljivo je da se posljednji paket (s rednim brojem 14) i posljednja potvrda (s rednim brojem 15) nisu izgubili.



Slika 2

Na slici *Slika 3* vidi se da je samo predajnik 1 slao svoje pakete jer je predajnik 2 prije završio. Na kraju i predajnik 1 završava i također je vidljivo da se ni njegov posljednji paket i posljednja potvrda nisu izgubili.



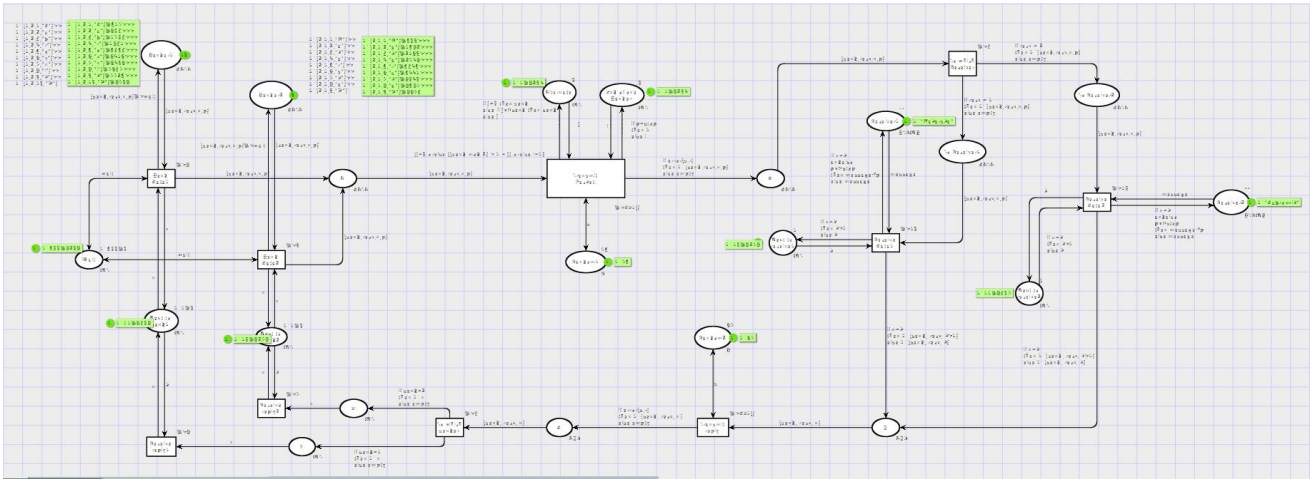
Slika 3

Slijede i rezultati verifikacije u kojima je vidljivo da nema grešaka i ne postoje nedohvatljiva stanja.

```
verification result:
spin -a  lab1.pml
gcc -DMEMLIM=1024 -O2 -DXUSAFE -DSAFETY -DNOCLAIM -w -o pan pan.c
./pan -m10000
Pid: 3410


(Spin Version 6.4.4 -- 1 November 2015)
        + Partial Order Reduction


Full statespace search for:
        never claim         - (not selected)
        assertion violations +
        cycle checks        - (disabled by -DSAFETY)
        invalid end states  +


State-vector 200 byte, depth reached 939, errors: 0
```

```
  496358 states, stored
   87891 states, matched
  584249 transitions (= stored+matched)
       0 atomic steps
hash conflicts:       765 (resolved)

Stats on memory usage (in Megabytes):
 107.927 equivalent memory usage for states (stored*(State-vector + overhead))
  64.710  actual memory usage for states (compression: 59.96%)
          state-vector as stored = 109 byte + 28 byte overhead
 128.000 memory used for hash table (-w24)
   0.534   memory used for DFS stack (-m10000)
 193.085 total actual memory usage



unreached in init
          (0 of 6 states)
unreached in proctype Sender
          (0 of 14 states)
unreached in proctype Network
          (0 of 91 states)
unreached in proctype Receiver
          (0 of 10 states)

pan: elapsed time 0.42 seconds
No errors found -- did you verify all claims?
```

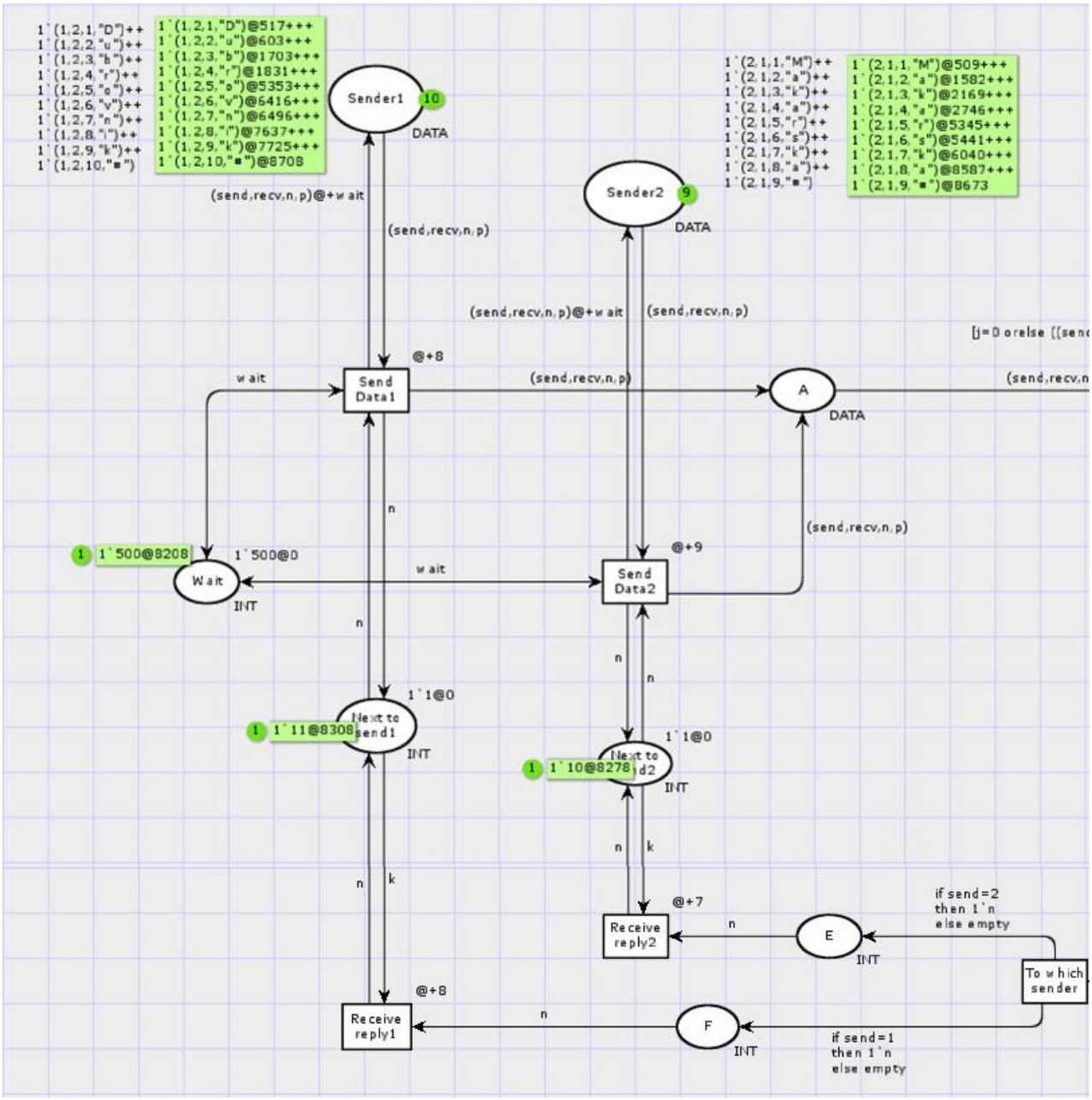
3. Model protokola pomoću Petrijeve mreže u alatu CPN Tools



Slika prikazuje protokol modeliran u alatu CPN Tools. Predajnik 1 (*Sender1*) šalje poruku „Dubrovnik“ prijamniku 2 (*Receiver2*), a predajnik 2 (*Sender2*) šalje poruku „Makarska“ prijamniku 1 (*Receiver1*).

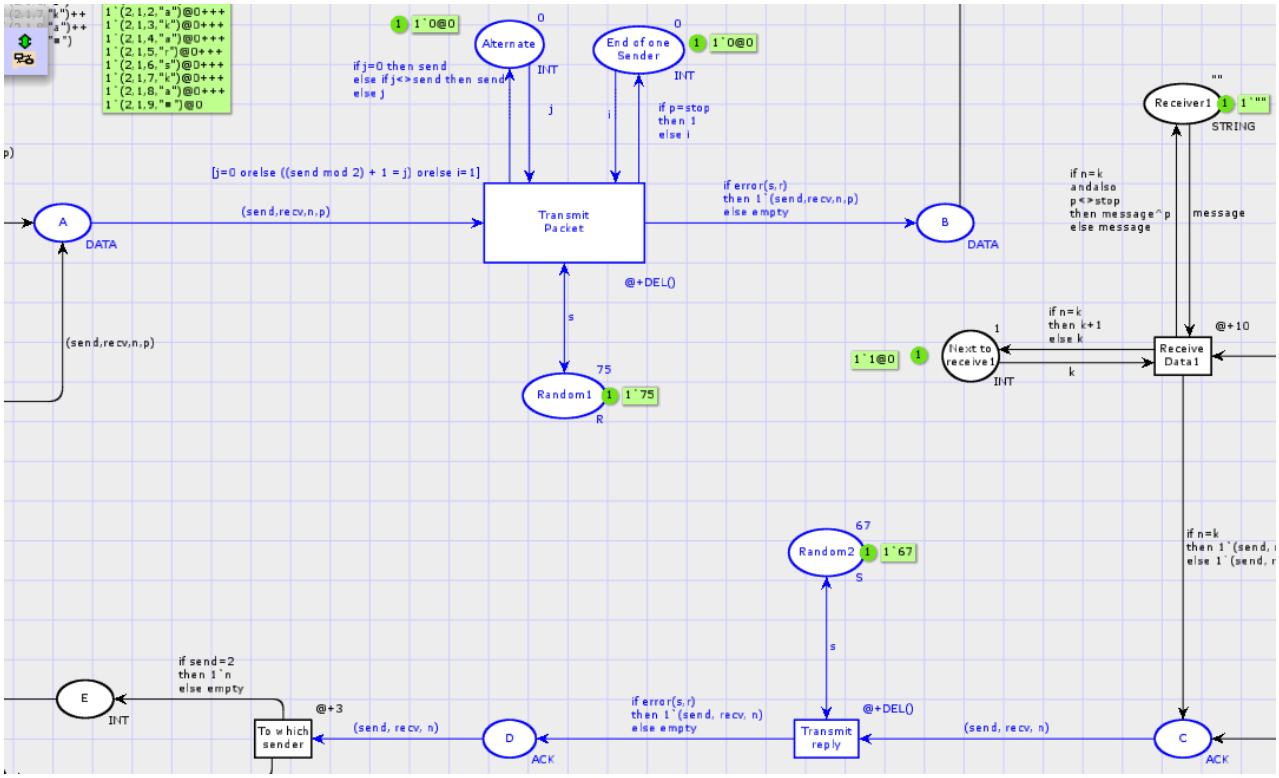
Sa slike je vidljiv dio mreže koji modelira predajnike. Svaki od njih ima:

- jedno mjesto u kojem su spremljeni paketi za slanje (*Sender1* i *Sender2*). Svaki token, odnosno svaka oznaka u tom mjestu sadrži ID predajnika *send* (1 ili 2), ID prijamnika *recv* (1 ili 2), redni broj paketa *n* i sadržaj paketa *p* (1 znak, tj. jedno slovo poruke) – (*send, recv, n, p*). Predajnik 1 šalje poruku „Dubrovnik“ prijamniku 2 (*recv* = 2), a predajnik 2 šalje poruku „Makarska“ prijamniku 1 (*recv* = 1)
- prijelaz (*Send Data1* i *Send Data2*) preko kojeg se paketi šalju u mrežu
- mjesto (*Next to send1* i *Next to send2*) u kojem se čuva redni broj paketa koji se sljedeći treba poslati. U početku je taj broj 1 (početna oznaka je 1) i povećava se svaki puta kad za poslani paket uspješno stigne potvrda
- prijelaz (*Receive reply1* i *Receive reply2*) koji iz mreže primaju potvrdu za poslani paket. Potvrda za pojedini paket je broj za jedan veći od rednog broja paketa i taj se broj dojavljuje mjestu *Next to send* koje onda zna koji se sljedeći paket treba poslati



Prijelaz *To which sender* iz primljene potvrde razlučuje kojem predajniku ona treba biti poslana (preko ID-a predajnika – *send*) te se ona preko posrednih mjesta *E* i *F* šalje do prijelaza *Receive reply1* i *Receive reply2*.

Mjesta i prijelazi sa slike označeni plavom bojom modeliraju komunikacijsku mrežu preko koje predajnici i prijemnici komuniciraju. Mjesto *A* prima paket i prosljeđuje ga do prijelaza *Transmit Packet*. Kako bi osigurala naizmjenično slanje paketa, na taj sam prijelaz dodala uvjet i spojila ga s dva mjesta.



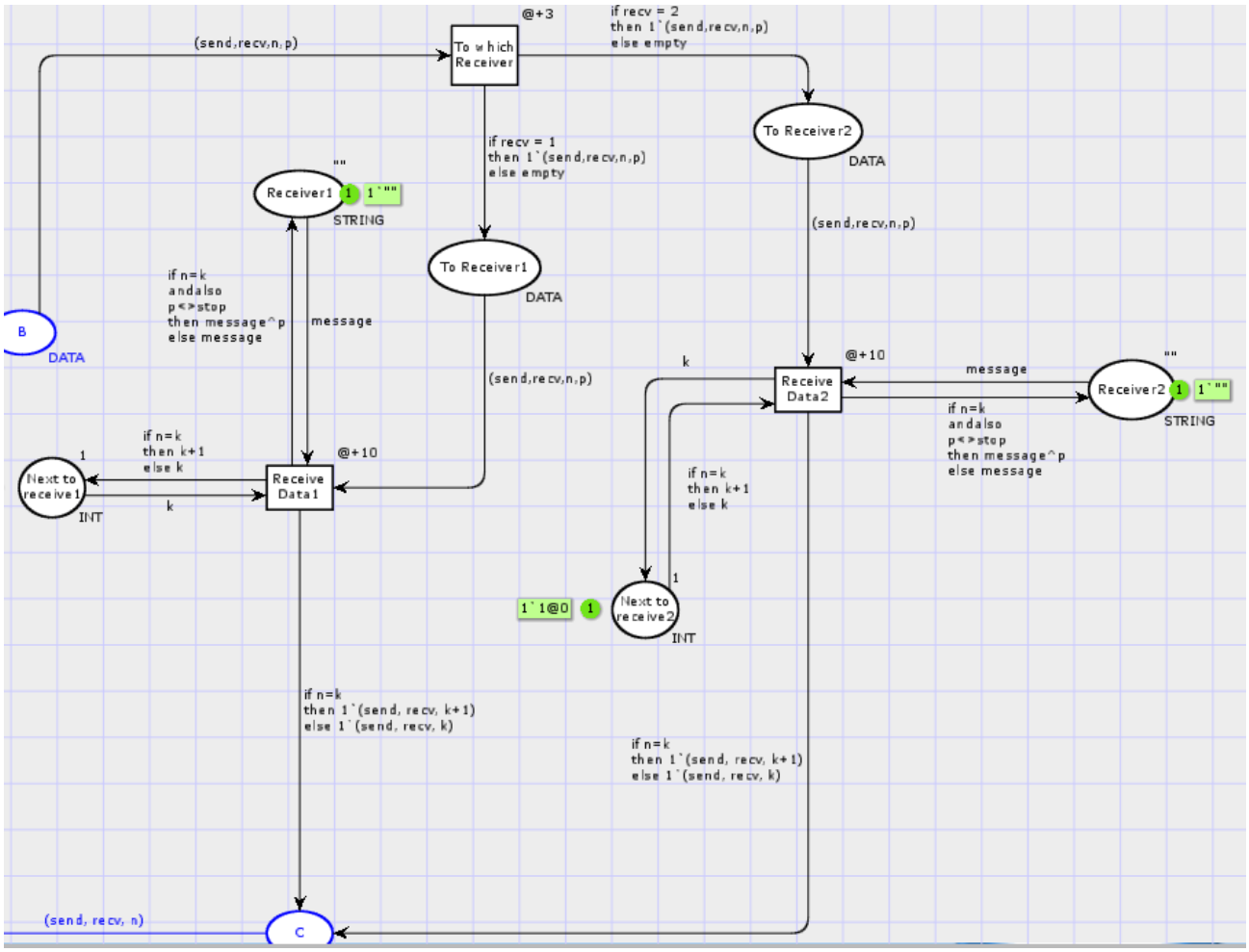
Mjesto *Alternate* pamti varijablu *j* koja označava ID predajnika čiji je paket prethodno poslan kroz prijelaz *Transmit Packet.* Inicijalno je *j*=0 kako bi se propustio paket od onog predajnika koji ga prvi pošalje. Nakon toga on postavlja *j* u svoj ID. Kada stigne paket od drugog predajnika provjerava se je li *j* jednak ID-u prvog predajnika i ako je, znači da je prethodno prvi predajnik poslao svoj paket i da je sad na redu drugi i njegov paket može proći. Kad jedan predajnik završi slanje svojih paketa, više se paketi ne šalju naizmjenično nego predajnik koji još nije dovršio šalje svoje pakete jedan za drugim. To je riješeno dodavanjem mjesta *End of one sender* koje pamti varijablu *i*. *i* je početno u 0, a kada prođe paket koji ima oznaku kraja poruke ('#'), *i* se postavlja u 1 i to označuje da više nema naizmjeničnog slanja paketa već se propuštaju svi paketi koji dolaze (jer dolaze samo paketi od onog predajnika koji nije dovršio slanje svoje poruke).

Još jedno mjesto spojeno na prijelaz *Transmit Packet* je *Random1*. To mjesto pamti varijablu *s* koja označava postotak uspješne prolaznosti paketa kroz mrežu do predajnika, odnosno 100 – s označava postotak gubitka paketa. To je izvedeno pomoću funkcije *error(s, r)* koja vraća istinitost izraza *r*<=*s*. *r* je broj u rasponu [1, 100] odabran nasumično. Ako je izraz *r*<=*s* istinit, uvjetom na luku prema mjestu B

osigurano je da paket dođe do tog mjesta, u suprotnom paket ne prolazi i on je izgubljen. Kako je vjerojatnost 0.25 (25%) da *r* bude veći od *s*, time je osigurano da se paket izgubi s vjerojatnošću 0.25.

Isti scenarij odvija se prije prosljeđivanja potvrde što obavlja prijelaz *Transmit reply*, promijenjena je samo vrijednost koju čuva mjesto *Random2* jer je vjerojatnost da se potvrda izgubi 0.33 – dakle postotak uspješnog prosljeđivanja potvrde je 67. Mjesta *C* i *D* su „posrednici“ između mreže i predajnika, odnosno između mreže i prijamnika.

Sljedeća slika prikazuje dio mreže koji modelira prijamnike.



U mjesta *Receiver1* i *Receiver2* spremaju se poruke pristigle od predajnika. Kako bi se one proslijedile pravom prijamniku (od *Sender1* do *Receiver2* i od *Sender2* do *Receiver2*), na izlaznom luku iz prijelaza *To which Receiver* uvjetom se provjerava ID prijamnika iz pristiglog paketa i ovisno koji je ID, paket se prosljeđuje do mjesta *To*

Receiver1 odnosno *To Receiver2* i dalje do prijelaza *Receive Data1* odnosno *Receiver Data2*.

Svaki prijamnik ima:

- mjesto koje pohranjuje dolazne pakete (*Receiver1* i *Receiver2*). Ono je inicijalno prazan string (“”) i sadržaji *p* dolaznih paketa se dodaju jedan na drugi. Prije toga se provjerava je li stigao ispravan paket
- prijelaz (*Receive Data1* i *Receive Data2*) koji prosljeđuje sadržaj paketa do mjesta *Receiver1* ili *Receiver2*
- mjesto (*Next to receive1* i *Next to receive2*) koje pamti varijablu *k* koja označava koji paket prijamnik treba sljedeći primiti. Početno je *k*=1 i povećava se svaki puta kad dođe ispravni paket. Dalje se ID predajnika, ID prijamnika i uvećani *k* šalju kao potvrda predajniku za uspješno primljen paket – (*send, recv, k*)

ANALIZA PROTOKOLA ALATOM CPN TOOLS

Izgenerirani izvještaj u CPN Toolsu slijedi u nastavku. Djelomičan je zbog postavljenih ograničenja kod naredbe *Calculate State Space* (*nodesstop*: 2000, *secsstop*: 300).

```
CPN Tools state space report for:
/cygdrive/E/Lorena/Faks/4. godina/Komunikacijski protokoli/LABOS.cpn
Report generated: Sun Nov 15 09:30:44 2015


 Statistics
------------------------------------------------------------------------
  State Space
    Nodes:  2022
    Arcs:   3376
    Secs:   0
    Status: Partial

  Scc Graph
    Nodes:  2022
```

```
   Arcs:   3376
   Secs:   0


 Boundedness Properties
---------------------------------------------------------------------------
 Best Integer Bounds
                    Upper     Lower
   PetriNet'A 1          2         0
   PetriNet'Alternate 1    1         1
   PetriNet'B 1          2         0
   PetriNet'C 1          0         0
   PetriNet'D 1          0         0
   PetriNet'E 1          0         0
   PetriNet'End_of_one_Sender 1
                    1         1
   PetriNet'F 1          0         0
   PetriNet'Next_to_receive1 1
                    1         1
   PetriNet'Next_to_receive2 1
                    1         1
   PetriNet'Next_to_send1 1
                    1         1
   PetriNet'Next_to_send2 1
                    1         1
   PetriNet'Random1 1      1         1
   PetriNet'Random2 1      1         1
   PetriNet'Receiver1 1    1         1
   PetriNet'Receiver2 1    1         1
   PetriNet'Sender1 1      10        10
   PetriNet'Sender2 1      9         9
   PetriNet'To_Receiver1 1 0         0
   PetriNet'To_Receiver2 1 1         0
   PetriNet'Wait 1         1         1


 Best Upper Multi-set Bounds
   PetriNet'A 1       1`(1,2,1,"D")++
1`(2,1,1,"M")
```

```
   PetriNet'Alternate 1
                1`0++
1`1++
1`2
   PetriNet'B 1       1`(1,2,1,"D")++
1`(2,1,1,"M")
   PetriNet'C 1       empty
   PetriNet'D 1       empty
   PetriNet'E 1       empty
   PetriNet'End_of_one_Sender 1
                1`0
   PetriNet'F 1       empty
   PetriNet'Next_to_receive1 1
                1`1
   PetriNet'Next_to_receive2 1
                1`1
   PetriNet'Next_to_send1 1
                1`1
   PetriNet'Next_to_send2 1
                1`1
   PetriNet'Random1 1  1`75
   PetriNet'Random2 1  1`67
   PetriNet'Receiver1 1
                1`""
   PetriNet'Receiver2 1
                1`""
   PetriNet'Sender1 1  1`(1,2,1,"D")++
1`(1,2,2,"u")++
1`(1,2,3,"b")++
1`(1,2,4,"r")++
1`(1,2,5,"o")++
1`(1,2,6,"v")++
1`(1,2,7,"n")++
1`(1,2,8,"i")++
1`(1,2,9,"k")++
1`(1,2,10,"#")
   PetriNet'Sender2 1  1`(2,1,1,"M")++
```

```
1`(2,1,2,"a")++
1`(2,1,3,"k")++
1`(2,1,4,"a")++
1`(2,1,5,"r")++
1`(2,1,6,"s")++
1`(2,1,7,"k")++
1`(2,1,8,"a")++
1`(2,1,9,"#")
   PetriNet'To_Receiver1 1
                empty
   PetriNet'To_Receiver2 1
                1`(1,2,1,"D")
   PetriNet'Wait 1    1`500

 Best Lower Multi-set Bounds
   PetriNet'A 1       empty
   PetriNet'Alternate 1
                empty
   PetriNet'B 1       empty
   PetriNet'C 1       empty
   PetriNet'D 1       empty
   PetriNet'E 1       empty
   PetriNet'End_of_one_Sender 1
                1`0
   PetriNet'F 1       empty
   PetriNet'Next_to_receive1 1
                1`1
   PetriNet'Next_to_receive2 1
                1`1
   PetriNet'Next_to_send1 1
                1`1
   PetriNet'Next_to_send2 1
                1`1
   PetriNet'Random1 1  1`75
   PetriNet'Random2 1  1`67
   PetriNet'Receiver1 1
                1`""
```

```
   PetriNet'Receiver2 1
                  1`""
   PetriNet'Sender1 1  1`(1,2,1,"D")++
1`(1,2,2,"u")++
1`(1,2,3,"b")++
1`(1,2,4,"r")++
1`(1,2,5,"o")++
1`(1,2,6,"v")++
1`(1,2,7,"n")++
1`(1,2,8,"i")++
1`(1,2,9,"k")++
1`(1,2,10,"#")
   PetriNet'Sender2 1  1`(2,1,1,"M")++
1`(2,1,2,"a")++
1`(2,1,3,"k")++
1`(2,1,4,"a")++
1`(2,1,5,"r")++
1`(2,1,6,"s")++
1`(2,1,7,"k")++
1`(2,1,8,"a")++
1`(2,1,9,"#")
   PetriNet'To_Receiver1 1
                  empty
   PetriNet'To_Receiver2 1
                  empty
   PetriNet'Wait 1    1`500

 Home Properties
--------------------------------------------------------------------------
  Home Markings
    Initial Marking is not a home marking

 Liveness Properties
--------------------------------------------------------------------------
  Dead Markings
    1929 [999,998,997,996,995,...]
```

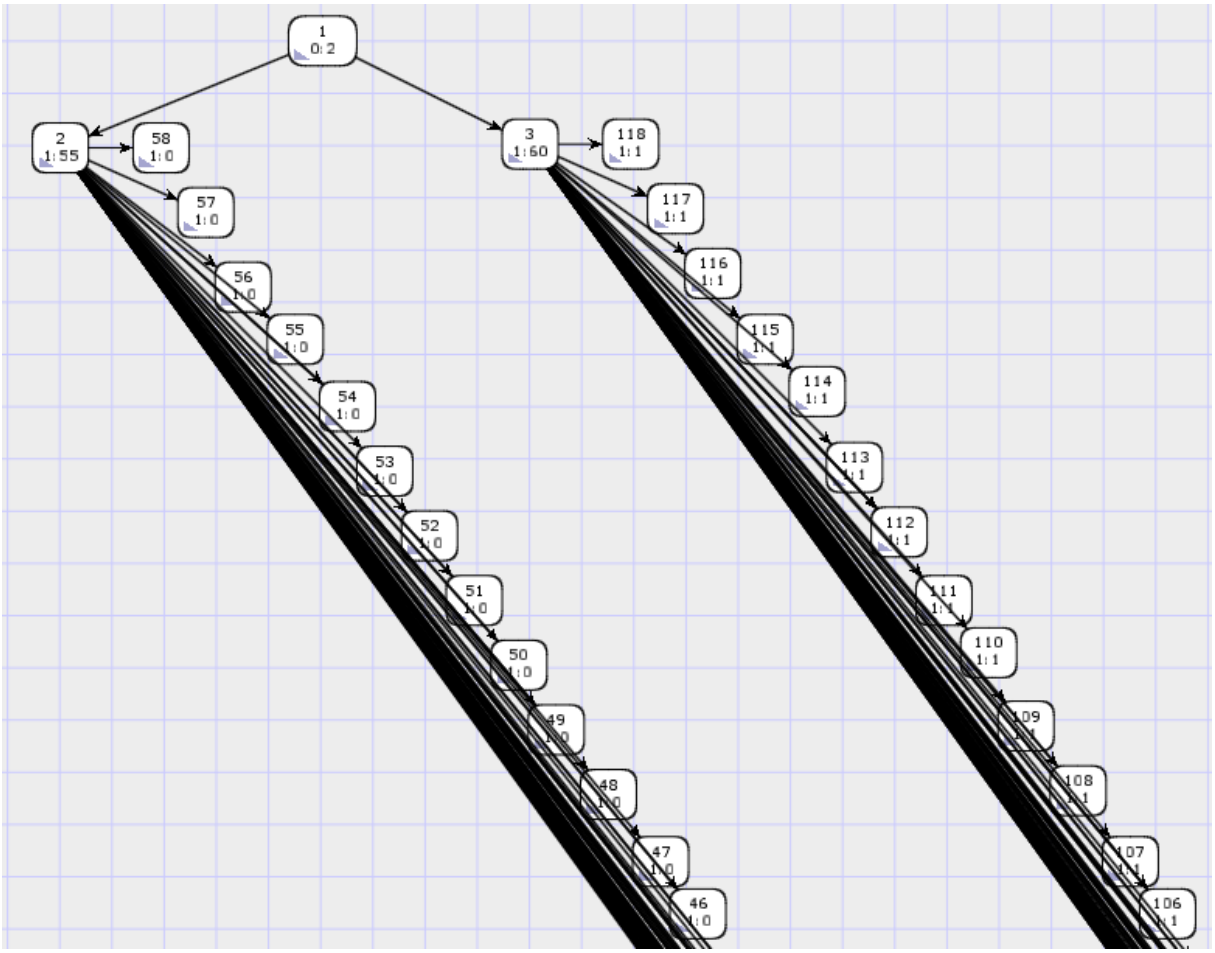
Live Transition Instances

None

Fairness Properties

No infinite occurrence sequences.

Na slici je prikazan djelomični graf stanja.



Svojstva mreže:

- **dostupnost** – dostupnost je odnos među stanjima koji se najbolje vidi iz grafa stanja. U mojoj su mreži sva stanja dostupna.
- **ograničenost** – ograničenost je najveći broj oznaka u mjestu na grafu stanja. Iz izvještaja je vidljivo da je ograničenost 10 zbog 10 oznaka u mjestu *Sender1* (D, u, b, r, o, v, n, i, k, #)

- **sigurnost** – svojstvo sigurnosti određuje da broj oznaka u svakome mjestu ne smije biti veći od jedan što u mojoj mreži ne vrijedi pa ona nije sigurna
- **aktivnost** – da bi mreža bila aktivna, u njoj ne smije postojati prijelaz koji se nikad ne izvodi ili stanje u kojem se ne može izvesti ni jedan prijelaz. Moja mreža nije aktivna jer nakon što se prenesu svi paketi od svakog predajnika ona staje s izvođenjem, više se ne izvede ni jedan prijelaz. U izvještaju je vidljivo da postoje mrtve oznake što dokazuje da mreža nije aktivna
- **reverzibilnost** – mreža je reverzibilna ako je početno stanje dostupno iz svakog stanja što u slučaju moje mreže ne vrijedi. Mreža nakon što se prenesu svi paketi od oba predajnika staje s izvođenjem i ne vraća se u početno stanje pa stoga nije reverzibilna
- **konfliktnost i simultanost** (mislim da sam tu izgubila bod) – dva su prijelaza konfliktna ako postoji stanje u kojem se oba mogu izvesti, ali izvedba jednog isključuje izvedbu drugog. U mojoj mreži to su npr. prijelazi *Receive reply1* i *Receive reply2* zbog uvjeta kod prijelaza *To which sender* koji pristiglu potvrdu šalje predajniku kojemu je namijenjena, dakle predajnici ne smiju dobiti potvrdu za paket koji nisu oni poslali. Prijelazi su simultani ako postoji stanje u kojem se oba mogu izvesti, a izvedba jednog ne utječe na izvedbu drugog. To su u mojoj mreži prijelazi *Send data1* i *Send data2* jer se paketi mogu istovremeno poslati, samo ne mogu istovremeno proći kroz prijelaz *Transmit Packet*.
- **perzistentnost** – perzistentnost je odsutnost konflikta, a kako u mojoj mreži konflikti postoje, ona nije perzistentna.

4. Usporedba

1. *Kako se osigurava sinkronizacija u Promeli, a kako u Petrijevoj mreži?*

Sinkronizaciju u Promeli osiguravaju prazni kanali. Tako kad neki proces pokuša staviti poruku na kanal ostaje zablokiran dok drugi proces ne pokuša čitati s tog kanala i obratno, kad neki proces pokuša čitati s kanala blokiran je dok drugi proces ne pokuša staviti poruku na kanal.

Sinkronizacija u Petrijevoj mreži osigurana je mjestima *Next to send1*, *Next to send2, Next to recv1* i *Next to recv2* koja pamte koje se poruke sljedeće trebaju primiti, odnosno poslati.

2. *Kako se simulira vrijeme u Promeli, a kako u Petrijevoj mreži?*

U Promeli se vrijeme simulira pomoću varijable *timeout,* a u Petrijevoj mreži pomoću oznaka *time stamps*, vremenskih oznaka. One su oblika @+*vrijeme*, gdje je *vrijeme* broj vremenskih jedinica koji predstavlja kašnjenje, odnosno koliko vremenskih jedinica treba čekati da izvede određena operacija.