Komunikacijski protokoli

Modeliranje komunikacijskih protokola programskim alatom Promela/Spin

Creative Commons

- slobodno smijete:
  - **dijeliti** — umnožavati, distribuirati i javnosti priopćavati djelo
  - **remiksirati** — prerađivati djelo
- pod sljedećim uvjetima:
  - **imenovanje**. Morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence (ali ne način koji bi sugerirao da Vi ili Vaše korištenje njegova djela imate njegovu izravnu podršku).
  - **nekomercijalno**. Ovo djelo ne smijete koristiti u komercijalne svrhe.
  - **dijeli pod istim uvjetima**. Ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, preradu možete distribuirati samo pod licencom koja je ista ili slična ovoj.

U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencne uvjete ovog djela. Najbolji način da to učinite je linkom na ovu internetsku stranicu.

Od svakog od gornjih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava.

Ništa u ovoj licenci ne narušava ili ograničava autorova moralna prava.

Tekst licencije preuzet je s http://creativecommons.org/.

Sadržaj predavanja

- Organizacija
  - Upute i rokovi
  - Preuzimanje alata
- Specifikacija, modeliranje i verifikacija protokola
- Programski alat Promela/Spin
  - Jezik Promela – detalji
- CPN Tools

Ocjenjivanje

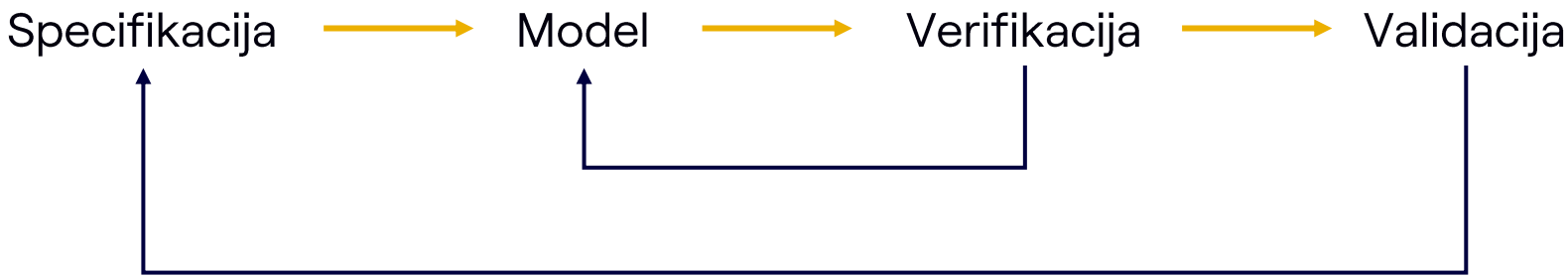
Komponente ocjene	Bodovi	
Sudjelovanje u nastavi	5	(aktivnost)
Laboratorijske vježbe	**10**	**(projektni zadatak)**
Domaća zadaća	10	
Međuispit	**30**	**(prvi dio gradiva)**
Završni ispit - pismeni	**30**	**(sve, s naglaskom na drugi dio)**
Završni ispit - usmeni	15	
ili ispitni rok	***75=60+15***	***(cijelo gradivo)***
Prolazna ocjena:	≥ 55	uz usmeni ispit ≥ 5, **labosi >0**

Projektni zadatak

- prvi dio – Promela
  - predaja do 28.10.
- drugi dio – CPN
  - predaja do 17.11.
- podjela zadataka – putem e-maila

- izvještaj – putem Moodlea
- gradivo laboratorijskih vježbi ulazi u MI, ZI i ispitne rokove

Modeliranje protokola



Promela/Spin

Promela/Spin

- http://spinroot.com/spin/Man/Manual.html
- Promela
  - Jezik za modeliranje komunikacijskih protokola i raspodijeljenih sustava
  - Modelira se komunikacija
  - Ostali detalji (npr. podaci koji se prenose) se zanemaruju
- Spin
  - Alat za verifikaciju raspodijeljenih sustava i komunikacijskih protokola
  - Provjeravaju se odsustvo zastoja (*deadlock*), nedefinirani primatelji poruka, nepotrebna stanja

Preuzimanje alata – Promela i Spin (1)

- Poteškoće s izvođenjem na Windowsima
- Virtualni stroj (Centos 6, 64-bitni) sa podešenim okružjem
  - http://public.tel.fer.hr/KomPro-LTI-LOI-Centos.ova
- Pokrenuti alatom za virtualizaciju
  - VirtualBox (http://www.oracle.com/technetwork/server-storage/virtualbox/downloads/index.html)
- Login: korisničko ime: **student LTI**, lozinka: **protokoli**
- Na desktopu otvoriti: ispin
- Primjer s predavanja: u direktoriju /home/student/Desktop
- Ostali primjeri: u direktoriju /home/student/Downloads/Spin/Examples

Preuzimanje alata – Promela i Spin (2)

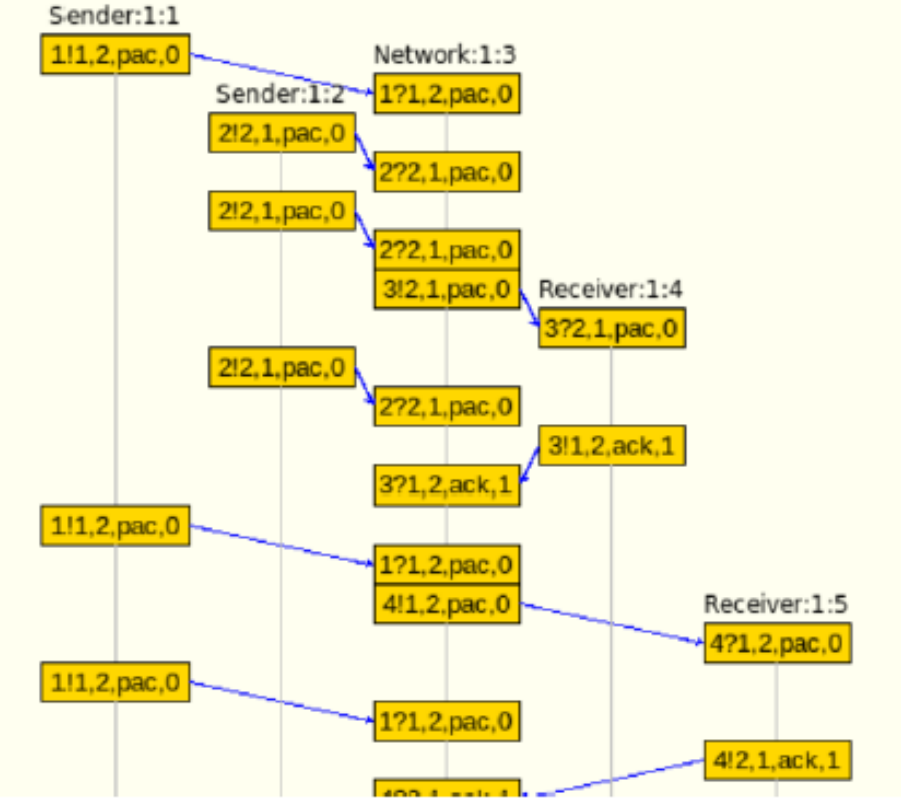
- Unix/Linux, Mac OS X
  - Ako koristite neki od ovih OS-ova, možete preuzeti alat spin i instalirati na vlastito računalo
  - Pratiti upute sa http://spinroot.com/spin/Man/README.html
  - Potrebno je imati instaliran gcc i Tcl/Tk

Promela

- Osnovni elementi jezika
  - Procesi – globalni
  - Kanali – globalni/lokalni
  - Varijable – globalni/lokalni

Spin

Redoslijed poruka



Verifikacija

unreached in init
(0 of 6 states)
unreached in proctype Sender
(0 of 15 states)
unreached in proctype Network
(0 of 73 states)
unreached in proctype Receiver
(0 of 11 states)

pan: elapsed time 0.23 seconds
No errors found -- did you verify all claims?

Tipovi podataka (1)

Ime	Veličina (bitovi)		Raspon
bit	1	nepredznačni	0..1
bool	1	nepredznačni	0..1
byte	8	nepredznačni	0..255
mtype	8	nepredznačni	0..255
short	16	predznačni	-2^15..2^15-1
int	32	predznačni	-2^32..2^32-1

Tipovi podataka (2)

```
mtype = {ack, err};
=
#define ack 2;
#define err 1;


byte state[N];
state[0] = state[4]+8*state[4*2/n];
```

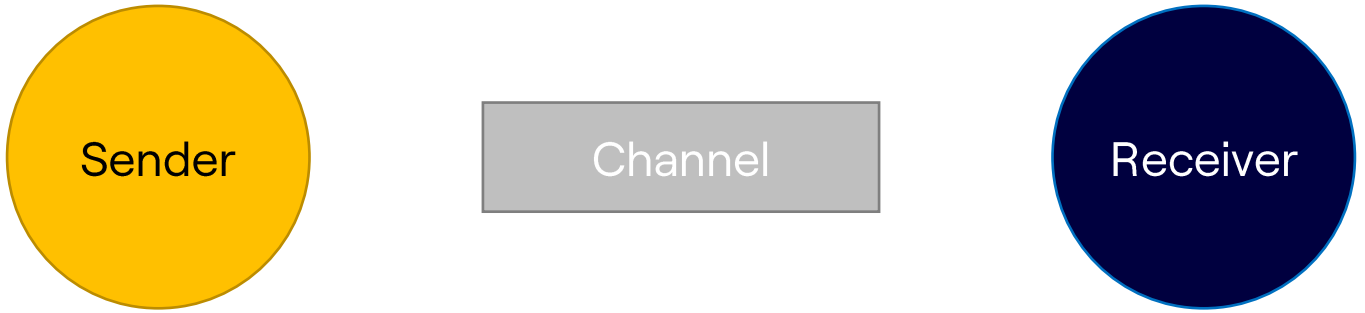
Izvodivost

- Nema razlike između uvjeta (*conditions*) i izraza (*statements*)
- Uvjeti (npr. `a==b`) se mogu koristiti kao izrazi
- Svaki izraz je izvodiv ili blokiran
- Izvodivost je osnovni način sinkronizacije

```
while (a!=b)
  skip
```

isto se postiže i izrazom: `(a==b)`

Procesi (1)



- Opisuju ponašanje kojim se mijenja stanje varijabli i komunikacijskih kanala

```
proctype A(){
  byte state;
  state=3
}
```

- Za odvajanje izraza može se koristiti `;` ili `->`

Procesi (2)

```
byte state = 2;

proctype A() {
  (state == 1) -> state=3
}

proctype B() {
  state = state - 1
}
```

Inicijalizacija procesa (1)

- Pri pokretanju Promela modela, izvršava se proces `init`

```
int state;
init {
  state=3;
  run A();
  run B();
}
```

Inicijalizacija procesa (2)

```
proctype A (byte state; short foo) {
  (state == 1) -> state = foo
}

init {
  run A(1,3)
}
```

Izvođenje procesa

- Čitanje i izmjena globalne varijable

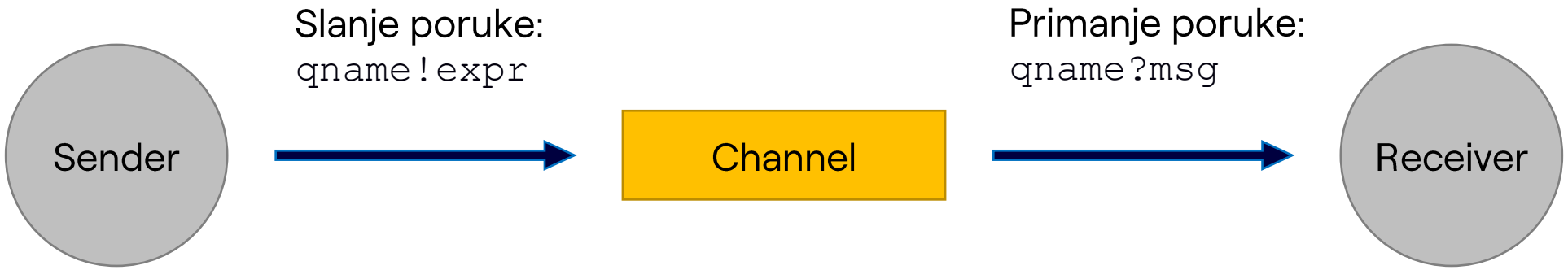
```
byte state = 1;
proctype A() {
  (state==1) -> state = state +1
}
proctype B() {
  (state==1) -> state = state -1
}
init {
  run A(); run B()
}
```

Naredba `atomic`

- Naredbe u bloku `atomic` se izvode slijedno i ništa ih ne može prekinuti

```
byte state = 1;
proctype A() {
  atomic {
       (state==1) -> state = state +1
  }
}
proctype B() {
  atomic {
       (state==1) -> state = state -1
  }
}
init {run A(); run B()}
```

Kanali



- Služe za modeliranje prijenosa poruka između procesa

```
chan qname = [16] of { short }
chan qname = [16] of { byte, int, chan, byte}
```

- Prijenos poruka po FIFO redoslijedu

Kanali (2)

- Moguće je slanje i primanje više podataka

```
qname!expr1,expr2,expr3
qname?var1,var2,var3
=
qname!expr1(expr2,expr3)
qname?var1(var2,var3)
```

```
#define msgtype 11;
chan name = [1] of {byte, byte, byte};
proctype A() {
  name!msgtype(12,13)
}
```

Kanali (3)

```
proctype A(chan q1) {
  chan q2;
  q1?q2;
  q2!123;
}
Proctype B(chan qforb) {
  int x;
  qforb?x;
  printf("x = %d\n", x)
}
Init {
  chan qname = [1] of {chan};
  chan qforb = [1] of {int};
  run A(qname);
  run B(qforb);
  qname!qforb
}
```

Kanali (4)

- Broj poruka u kanalu: `len(qname)`
- Provjera nalazi li se u kanalu podatak: `qname?[var]`

  1 – ako se nalazi

  0 – ako se ne nalazi

- Sinkrona komunikacija – kanal ne pohranjuje podatke

```
chan qname = [0] of {byte}
```

- Asinkrona komunikacija – kanal može pohraniti podatke

```
chan qname = [N] of {byte}
```

Komunikacija između procesa – pitanja 1

```
chan name = [0] of {int, int, int};

proctype A() {
  name!1,2,4;
  name!8,16,32;
}
proctype B() {
  int state1,state2, state3;
  name?state1, state2, state3
}
init {
  atomic {
        run A();
        run B();
  }
}
```

Komunikacija između procesa – pitanja 1

```
chan name = [0] of {int, int, int};

proctype A() {
  name!1,2,4;
  name!8,16,32;
}
proctype B() {
  int state1,state2, state3;
  name?state1, state2, state3
}
init {
  atomic {
        run A();
        run B();
  }
}
```

1.1) Što će se dogoditi pri izvršavanju ovog programa?
1.2) Što se nalazi na kanalu?
1.3) Jesu li procesi završili/ostali blokirani?
1.4) Koje su vrijednosti varijabli?

Sinkrona komunikacija!

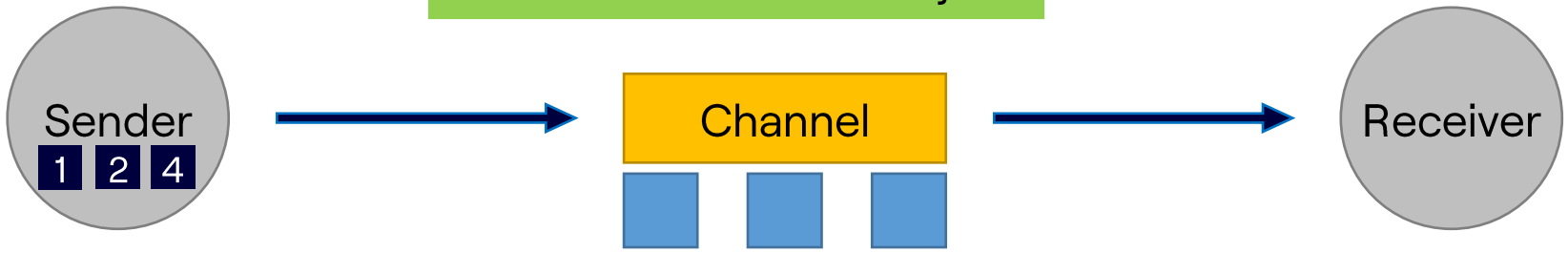
Komunikacija između procesa – pitanja 2

```
chan name = [1] of {int, int, int};

proctype A() {
  name!1,2,4;
  name!8,16,32;
}
proctype B() {
  int state1,state2, state3;
  name?state1, state2, state3
}
init {
  atomic {
        run A();
        run B();
  }
}
```

2.1) Što bi se dogodilo kad bi veličina kanala bila 1?

Asinkrona komunikacija!



Komunikacija između procesa – pitanja 2

```
chan name = [1] of {int, int, int};

proctype A() {
  name!1,2,4;
  name!8,16,32;
}
proctype B() {
  int state1,state2, state3;
  name?state1, state2, state3
}
init {
  atomic {
        run A();
        run B();
  }
}
```

Na kanalu!

2.1) Što bi se dogodilo kad bi veličina kanala bila 1?

Asinkrona komunikacija!

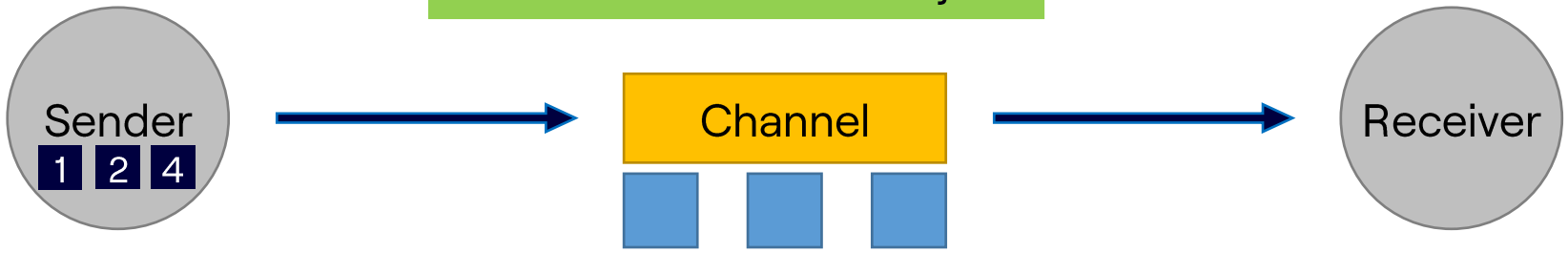
Komunikacija između procesa – pitanja 3

```
chan name = [1] of {int, int, int};

proctype A() {
  name!1,2,4;
  name!8,16,32;
}
proctype B() {
  int state1,state2, state3;
  name?[state1, state2, state3] ->
  name?state1, state2, state3
}
init {
  atomic {
        run A();
        run B();
  }
}
```

3.1) Što će se dogoditi pri izvršavanju ovog programa?

Asinkrona komunikacija!



Komunikacija između procesa – pitanja 3

```
chan name = [1] of {int, int, int};

proctype A() {
  name!1,2,4;
  name!8,16,32;
}
proctype B() {
  int state1,state2, state3;
  name?[state1, state2, state3] ->
  name?state1, state2, state3
}
init {
  atomic {
        run A();
        run B();
  }
}
```

3.1) Što će se dogoditi pri izvršavanju ovog programa?

Asinkrona komunikacija!

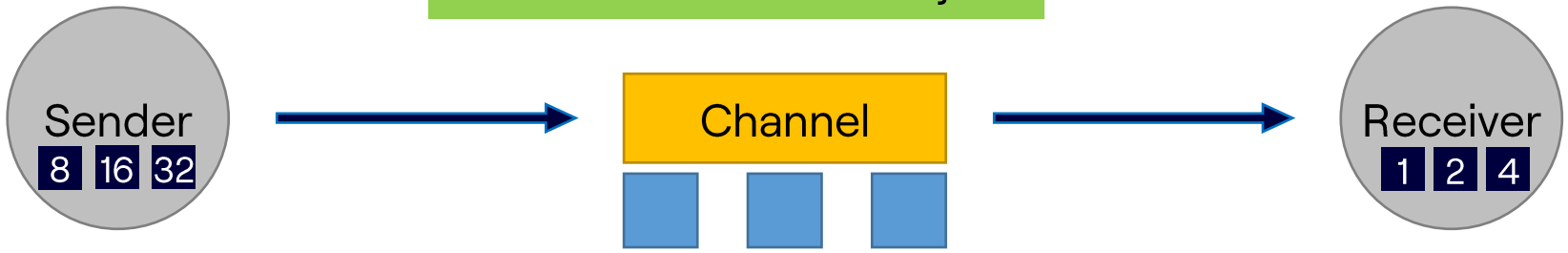
Komunikacija između procesa – pitanja 3

```
chan name = [1] of {int, int, int};

proctype A() {
  name!1,2,4;
  name!8,16,32;
}
proctype B() {
  int state1,state2, state3;
  name?[state1, state2, state3] ->
  name?state1, state2, state3
}
init {
  atomic {
        run A();
        run B();
  }
}
```

3.1) Što će se dogoditi pri izvršavanju ovog programa?

Asinkrona komunikacija!



Komunikacija između procesa – pitanja 3

```
chan name = [1] of {int, int, int};

proctype A() {
  name!1,2,4;
  name!8,16,32;
}
proctype B() {
  int state1,state2, state3;
  name?[state1, state2, state3] ->
  name?state1, state2, state3
}
init {
  atomic {
       run A();
       run B();
  }
}
```

Na kanalu!

Provjera

3.1) Što će se dogoditi pri izvršavanju ovog programa?

Asinkrona komunikacija!

Kontrola toka

- Slijed izraza

```
(a==b) -> state=3
```

- Paralelno izvođenje procesa
- Blok `atomic`

- Odabir slučaja (naredba `if`)
- Ponavljanje (petlje)
- Uvjetni skokovi

Odabir slučaja (1)

```
if
:: (a != b) -> option1
:: (a == b) -> option2
fi
```

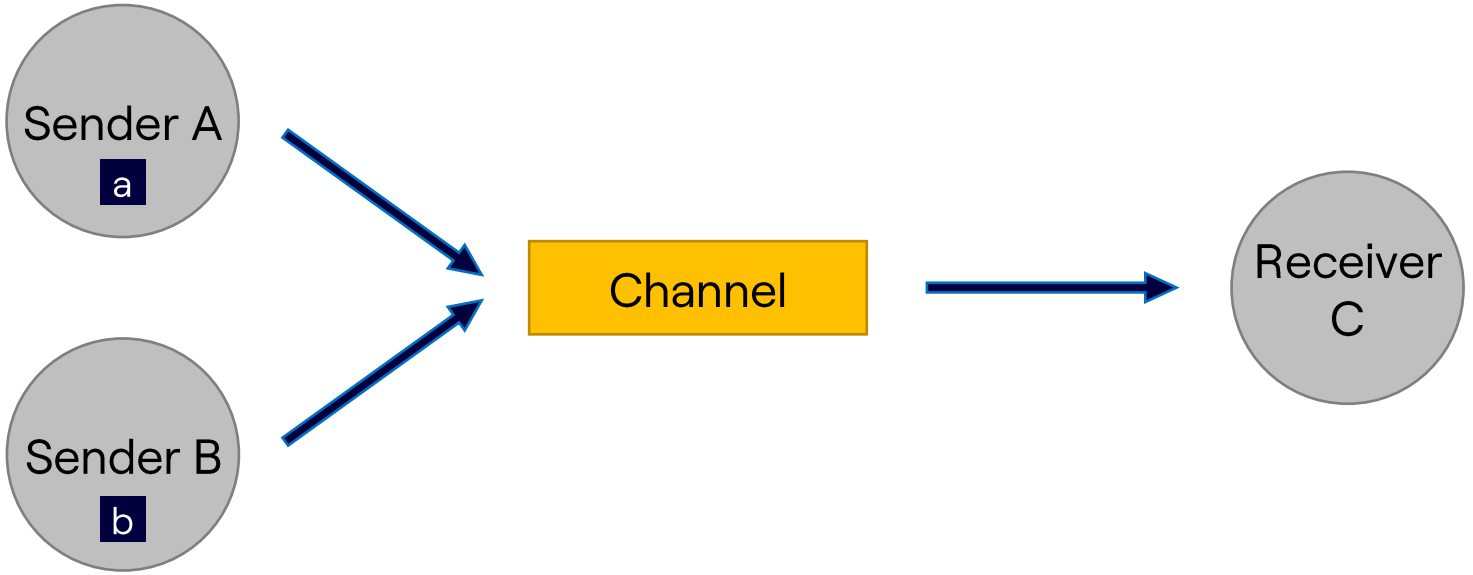
- Izvršava se samo jedan slijed naredbi
- Ako se ne može izvesti niti jedan slijed naredbi, proces je blokiran

```
if
:: (a == 3) -> option1
:: (b == 5) -> option2
fi
```

Odabir slučaja (2)

```
#define a 1
#define b 2
chan ch = [0] of { byte };

proctype A() { ch!a }
proctype B() { ch!b }
proctype C() {
  if
  :: ch?a
  :: ch?b
  fi
}
init { atomic { run A(); run B(); run C() }}
```



Odabir slučaja (3)

- Ako se može izvesti više sljedova naredbi, onaj koji će se izvesti se bira slučajnim odabirom

```
byte count;

proctype counter() {
  if
  :: count = count + 1
  :: count = count - 1
  fi
}
```

```
byte count;

proctype counter() {
  if
  :: count = count + 1
  :: count = count - 1
  :: count = 1
  :: count = 0
  fi
}
```

Petlje

```
byte count;

proctype counter() {
  do
  :: count = count + 1
  :: count = count - 1
  :: (count == 0) -> break
  od
}
```

Petlje – izlaz (1)

```
byte count;

proctype counter() {
  do
       :: (count!=0) ->
              if
                     :: count = count + 1
                     :: count = count - 1
              fi
       :: (count==0) -> break
  od
}
```

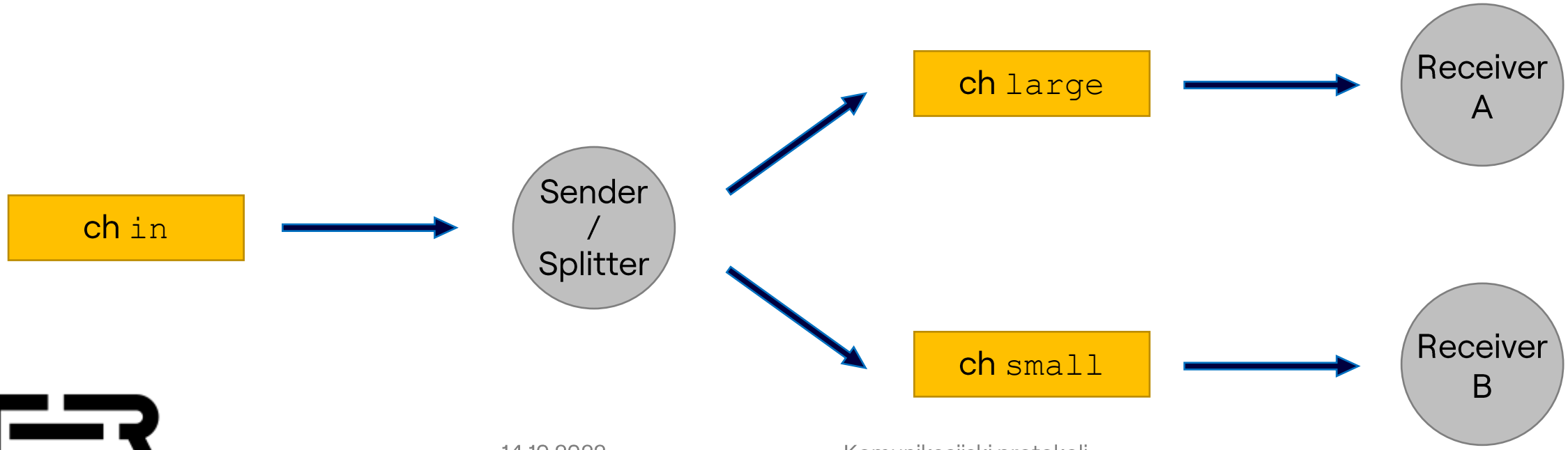
Petlje – izlaz (2)

```
byte count;

proctype counter() {
     do
       :: (count!=0) ->
            if
                 :: count = count + 1
                 :: count = count - 1
            fi
       :: (count==0) -> goto done
     od
done:
     skip
}
```

Primjer 1

- Napisati program koji filtrira poruke u kanalu
  - Vrijednosti veće od 128 se prenose kanalom `large` do primatelja A
  - Vrijednosti manje od 128 se prenose kanalom `small` do primatelja B
  - Nakon što primatelji prime poruke, ponovno ih šalju ispočetka



Primjer 1 – pokretanje procesa

```
#define N 128
#define size 16
chan in = [size] of { short };
chan large = [size] of { short };
chan small = [size] of { short };

proctype A() {…}
proctype B() {…}
proctype split() {…}

init {
        in!345; in!12; in!677; in!32; in!0;
        run split();
        run A();
        run B()
}
```

Primjer 1 – procesi A i B

```
proctype A() {
      short cargo;

      do
      :: large?cargo; in!cargo
      od
}
```

```
proctype B() {
      short cargo;

      do
      :: small?cargo -> in!cargo
      od
}
```

Primjer 1 – proces koji filtrira poruke

```
proctype split() {
        short cargo;
        do
        :: in?cargo ->
                if
                :: (cargo >= N) -> large!cargo
                :: (cargo < N) -> small!cargo
                fi
        od
}
```

Primjer 2 (1)

Sinkronizacija procesa pomoću semafora

```
chan sema = [0] of { bit }
proctype dijkstra() {

        do
        :: sema!0
        :: sema?1
        od
}

proctype user() {
        do
        ::  sema?0; /* critical section */
            sema!1; /* non-critical section */
        od
}
```

Primjer 2 (2)

Sinkronizacija procesa pomoću semafora

```
chan sema = [0] of { bit }
proctype dijkstra() {
	byte count = 1;
	do
	:: (count == 1) -> sema!0; count = 0
	:: (count == 0) -> sema?1; count = 1
	od
}

proctype user() {
	do
	::  sema?0; /* critical section */
	    sema!1; /* non-critical section */
	od
}
```

Prekid procesa

Istek vremena

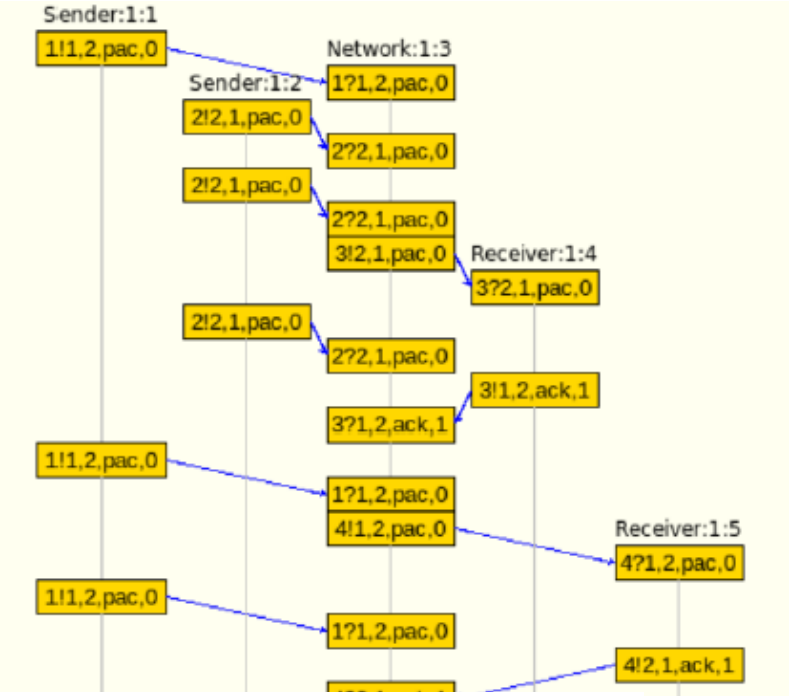
```
proctype watchdog() {
      do
      :: ch?a
      :: timeout -> guard!reset
      od
}
```

Izvršava se kad ostale naredbe u `do` bloku nisu izvršive. Samostalno treba implementirati što činiti u takvom slučaju!

`timer`, `set`, `expire`...

Spin

- Edit/View
- Simulate/Replay
  - Izvršiti simulaciju specificiranog protokola
  - Isporbati različite *seedove*
    - polje *Random, with seed*
  - Staviti sliku dijela s razmjenom poruka u izvještaj
- Verification
  - Opcija *Run*
  - Provjeriti greške i nedostignuta stanja




```
unreached in init
        (0 of 6 states)
unreached in proctype Sender
        (0 of 15 states)
unreached in proctype Network
        (0 of 73 states)
unreached in proctype Receiver
        (0 of 11 states)

pan: elapsed time 0.23 seconds
No errors found -- did you verify all claims?
```

CPN Tools

CPN Tools

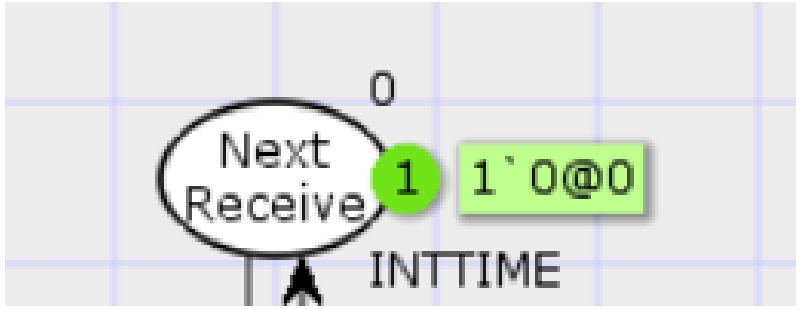
- Alat za modeliranje konkurentnih sustava korištenjem Petrijevih mreža
  - Mreža mjesta i prijelaza
- Preuzeti sa web-stranice http://cpntools.org/download
- Zadaci
  - Modeliranje komunikacijske mreže pomoću Petrijeve mreže
    - Opisati mjesta i funkcije prijelaza
  - Analiza Petrijeve mreže alatom CPN Tools (*State Space Tool*)
  - Određivanje svih svojstava mreže (s predavanja)
  - Graf stanja (djelomični)

CPN Tools – osnove

- Većina opcija – desni klik miša
- *Tool box*
  - *Net* – opcije oko spremanja i naziva
  - *Simulation*
  - *Hierarchy*
  - *State space* – verifikacija mreže, određivanje svojstava
- Standard priorities - konstante
- Standard declarations
  - Skup (boja) – skup kojem pripadaju oznake u stanjima
  - Varijable
    - Prenose se granama
    - Moraju odgovarati skupu boja u stanju iz kojeg/u koje prelaze

Mjesta

- Jedinstveni naziv
- Skup oznaka
  - INT – sve integer *vrijednosti*
  - Proizvoljno definirane vrijednosti (1,2,3)
  - Složeni skupovi (jedna oznaka: 1, poruka – INT, STRING)
- Početna oznaka
- Uređivanje zapisa – klik na mjesto, pomicanje tipkom TAB

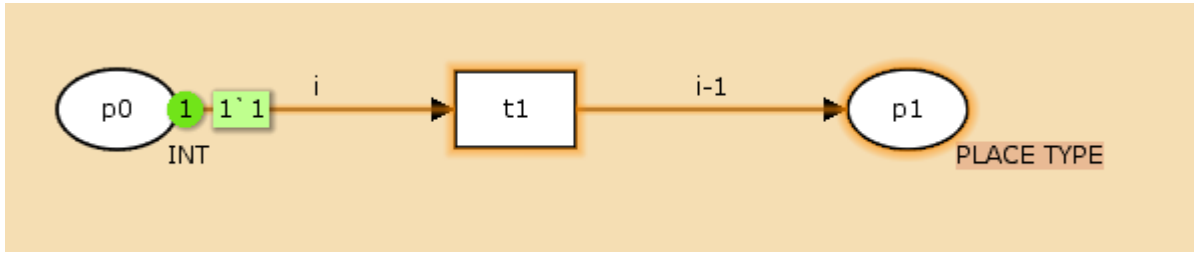


Mjesta (2)

- PLACE TYPE
  - grana koja vodi u mjesto mora imati odgovarajući COLSET
  - direktno na grani je moguće mijenjanje vrijednosti

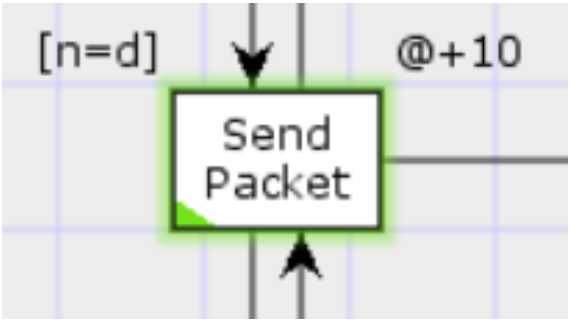
- varijabla mora biti definirana nakon COLSET-a

```
colset INT = int;
var i: INT;
```



Prijelazi

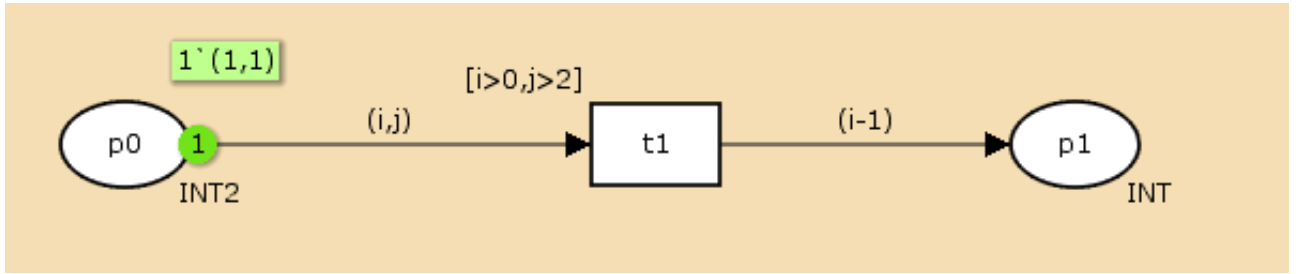
- Jedinstveni naziv
- Uvjet izvršavanja -> mora biti zadovoljen da bi se prijelaz izveo
- Vremenska oznaka @
  - Za simulacije u ovisnosti u vremenu
  - Definira se trajanje prijelaza
- Prioritet
- Kod
  - Mogućnost definiranja određenih akcija
  - Alternativa: definiranje akcija na granama
- Uređivanje zapisa – klik na prijelaz, pomicanje tipkom TAB



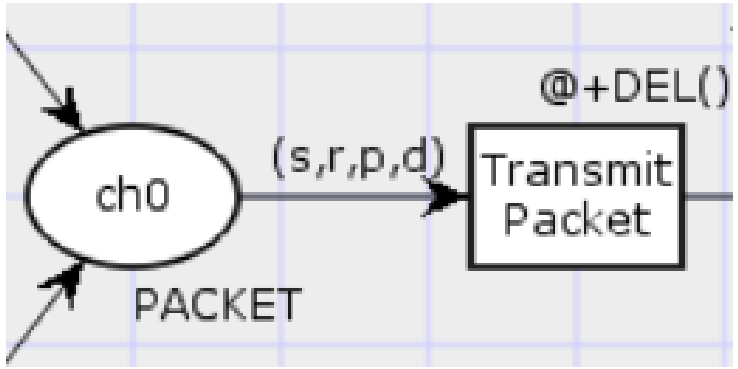
Prijelazi (2)

- moguće više uvjeta na istom prijelazu

```
colset INT = int;
colset INT2 = product INT*INT;
var i,j: INT;
```



Grane

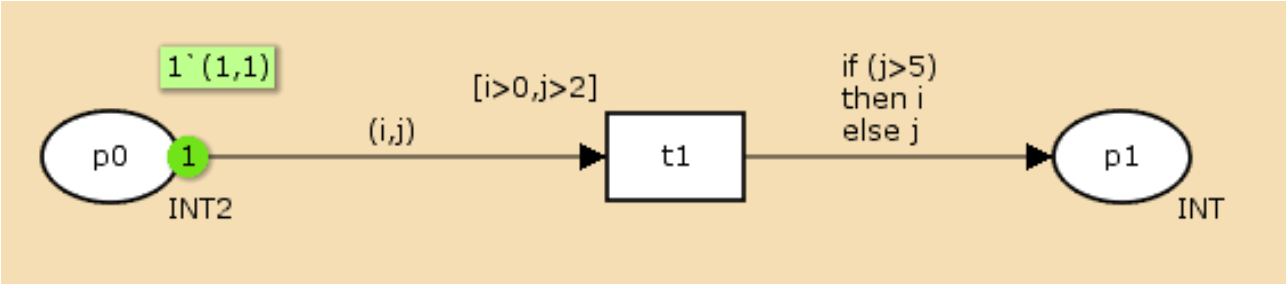


- Varijable
- Izrazi
- Kašnjenja
- Varijable i rezultati izraza moraju biti u skupu koji podržava mjesto spojeno na granu!
- Na granu se može postaviti funkcija koja uzrokuje gubitkom poruke/potvrde

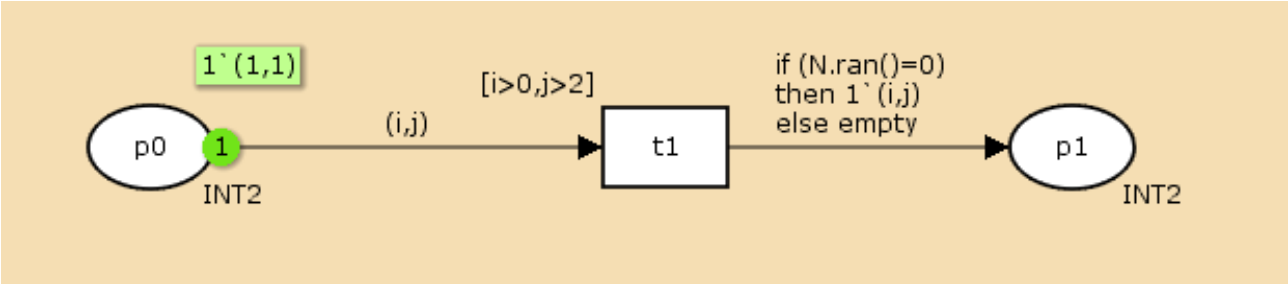
Grane (2)

- `if - else` na grani

```
colset INT = int;
colset INT2 = product INT*INT;
var i,j: INT;
```



- `if - else` s gubitkom



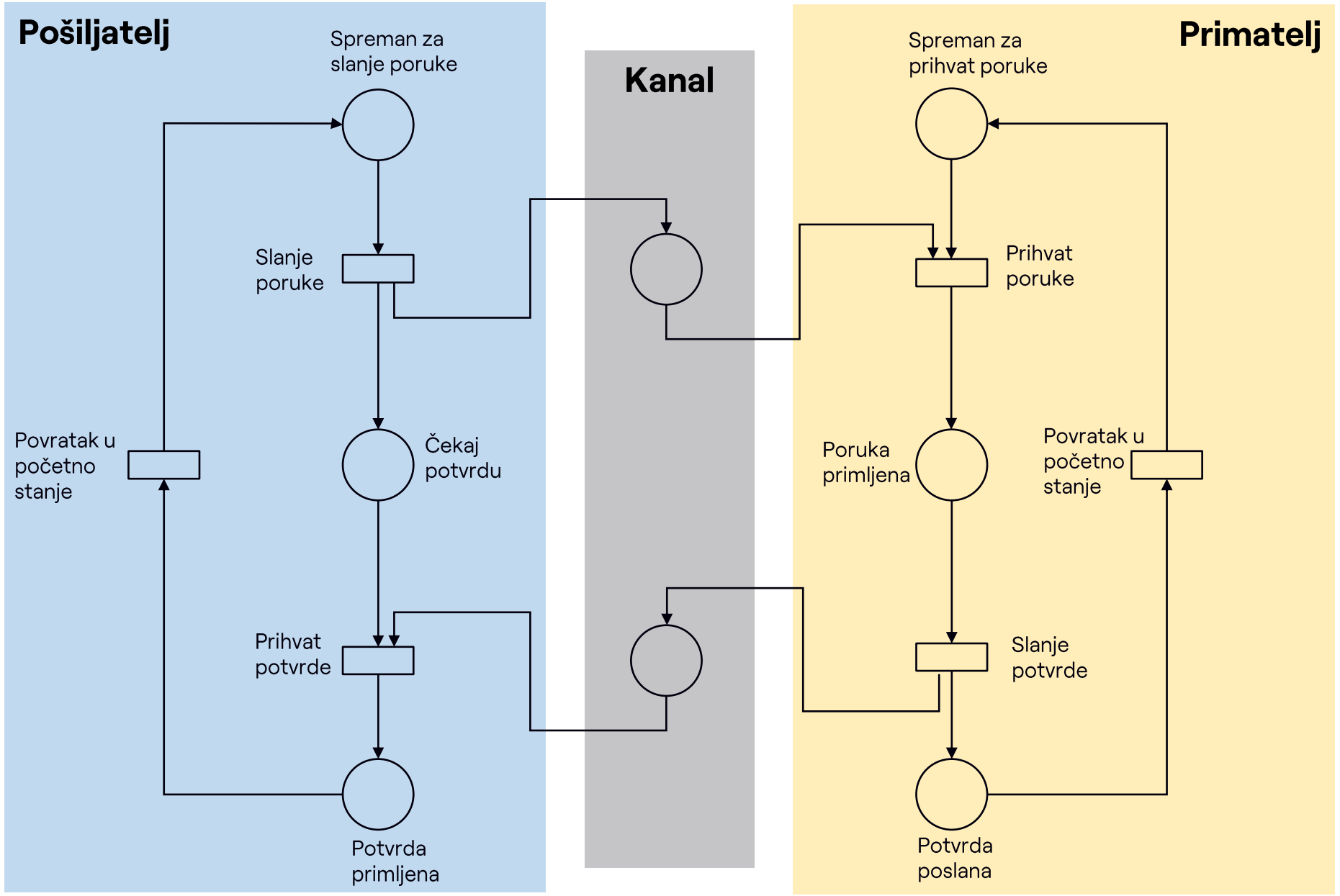

```
colset N = int with 0..1;
```

Primjeri

- CPN Tools
  - Osnovni model protokola
  - Gubitak – 1 predajnik, 1 prijamnik
  - CPN Tools/Sample CPN Models/Timed Protocol

Verifikacija – svojstva Petrijeve mreže

- Ograničenja
  - Best Integer Bounds
- Domaće stanje – stanje u koje se moguće vratiti - reverzibilnost
- Aktivnost
  - Mrtva stanja – stanja iz kojih nema izlaza
  - Mrtvi prijelazi – prijelazi koji se ne mogu izvršiti
  - Izvršivi prijelazi
- Pravednost (Fairness)
  - Koliko se često izvodi određeni slijed prijelaza


Pošiljatelj
Spreman za slanje poruke
Slanje poruke
Čekaj potvrdu
Povratak u početno stanje
Prihvat potvrde
Potvrda primljena
Kanal
Primatelj
Spreman za prihvat poruke
Prihvat poruke
Poruka primljena
Povratak u početno stanje
Slanje potvrde
Potvrda poslana

Demo – Spin i CPN Tools

- https://www.youtube.com/playlist?list=PLRFTXn5ZdqaNjeUKXe-qk_X41bL37rKqn