# Poglavje 4

# Barvne Petrijeve mreže

V predhodnem poglavju o Petrijevih mrežah smo ugotovili, da dinamiko v opazovani Petrijevi mreži odslikuje premikanje žetonov po njenih pogojih, kar smo si interpretirali kot spreminjanje stanj opazovane Petrijeve mreže. Pomeni žetonov so bili v podanih primerih različni in predmet subjektivne interpretacije opazovalca. Tako so žetoni v različnih zgledih Petrijevih mrež predstavljali *zahteve*, porabljive ali neporabljive *resurse*, *sprožilce dogajanja*, *programske pogoje*, *števce*, *čuvaje*, *sporočila*, *podatkovne* in *kontrolne pakete* itd. V pričujočem poglavju obstoječe Petrijeve mreže razširimo na *barvne Petrijeve mreže*, v katerih je pomen posameznega žetona enolično določen.

*Barvne Petrijeve mreže* (angl. *coloured Petri nets*) predstavljajo razširitev običajnih Petrijevih mrež, ki smo jih dodobra spoznali v predhodnem poglavju. Podobno kot običajne, tudi barvne Petrijeve mreže predstavljajo grafično orientiran jezik, ki ga uporabljamo za potrebe načrtovanja, specificiranja, modeliranja in verifikacije dinamičnih sistemov [16], [17]. Uporaben je predvsem na področjih komunikacij, sinhronizacije procesov in v sistemih, kjer prihaja do deljenja resursov. Na področju računalniških komunikacij jih uporabljamo predvsem pri snovanju in verifikaciji pravilnosti delovanja komunikacijskih protokolov.

## 4.1 Definicija barvnih Petrijevih mrež

V običajnih Petrijevih mrežah smo spoznali vlogo žetonov, pri čemer med njimi glede na formalni zapis mreže pomensko nismo razlikovali. Navkljub temu, da so žetoni predstavljali „čuvaje“, „vrednosti števcev“, „posle“, „resurse“, „pakete“ itd., so bili njihovi pomeni neformalizirani in kot taki le rezultat naše vsakokratne subjektivne interpretacije.

Barvne Petrijeve mreže omogočajo formalno specifikacijo *pomena* oziroma *tipa žetona*, ki ju dosežemo z dodeljevanjem enolične deklaracije posameznemu tipu žetona. Pod pojmom enolične deklaracije tipa žetona imamo v mislih dodelitev *sestavljene podatkovne strukture* oziroma njegove *barve* (angl. *token colour*)

posameznemu tipu žetona. Na ta način pridobimo možnost enoličnega razlikovanja med posameznimi tipi žetonov. S takšno razširitvijo žeton omogoča prenos vrednostno inicializiranih spremenljivk po barvni Petrijevi mreži. V ta namen moramo v fazi gradnje modela deklarirati *tipe žetonov* oziroma definirati njim ustrezne podatkovne tipe - *barvne nabore*. Posamezni žeton v barvni Petrijevi mreži tako pridobi svojo barvo (podatkovno strukturo), poimenujemo pa ga za *barvni žeton*.

V nadaljevanju navedemo poenostavljeno definicijo barvnih Petrijevih mrež povzeto po viru [18], pri čemer v definiciji dopustimo tudi možnost časovne opredelitve trajanja akcij, ki jo navedeni vir ne navaja.

**Definicija 16** *Barvna Petrijeva mreža je definirana kot osmerček* $C = (P, T, I, O, o(t_0), D, C, S)$*, pri čemer* $P$ *predstavlja končno množico pogojev,* $T$ *končno množico akcij,* $I$ *vhodno in* $O$ *izhodno preslikavo. Množici* $P$ *in* $T$ *sta si tuji* ($P \cap T = \emptyset$)*. Vektor* $o(t_0)$ *predstavlja začetno označitev mreže,* $D$ *pa vektor časovnih trajanj posameznih akcij. Velja izraz*

$$C : P \to S, \tag{4.1}$$

*kjer* $S$ *predstavlja množico vseh barvnih naborov,* $C$ *pa barvno preslikavo. Slednja posameznemu pogoju določi zanj potrebne barvne nabore.*

Množica vseh barvnih naborov vsebuje vse tiste sestavljene podatkovne strukture, ki jih potrebujemo za opredelitev modela opazovanega sistema z barvno Petrijevo mrežo.

Pojem žetona bomo v nadaljevanju pričujočega poglavja zamenjali s pojmom **barvnega žetona**. Za izvedbo opazovane akcije v barvni Petrijevi mreži mora biti izpolnjen pogoj, da se v vseh pogojih, iz katerih vstopajo povezave v opazovano akcijo, nahajajo barvni žetoni, ki se ujemajo po vrednostih enakih spremenljivk. Če navedeni pogoj za opazovano akcijo ni izpolnjen, le te ne moremo izvesti. Neenake vrednosti istih spremenljivk barvnih žetonov v pogojih, iz katerih vstopajo povezave v opazovano akcijo, tako izvajanje te akcije onemogočijo.

V nadaljevanju si bomo ogledali dva zgleda primerov modelov komunikacijskih protokolov ponazorjena z barvnimi Petrijevimi mrežami.

## 4.2 Model enosmernega oddajno sprejemnega protokola z neidealno prenosno potjo

Predpostavimo, da imamo opravka z oddajno sprejemnim protokolom, opisanim v zadnjem razdelku prejšnjega poglavja, ki poteka po neidealnem kanalu. V namene odpravljanja izgub in okvar paketov protokol vsebuje potrjevanje paketov. Predpostavimo, da pakete protokol oštevilčuje z zaporednimi številkami

in ne na alternirajoč bitni način, kot je bilo predstavljeno v zgledu iz prejšnjega poglavja.

Pred samo zasnovo modela protokola moramo definirati tipe barvnih žetonov oziroma še pred tem osnovne podatkovne strukture, ki jih bomo dodeljevali barvnim žetonom, s katerimi bomo v nadaljevanju definirali delovanje opisanega protokola. Deklaracijo podatkovnih struktur zapišemo z deklaracijskimi izrazi v izpisu 4.1.

```
1 p: CHAR[8]; % podatkovna vsebina paketa
2 n: INTEGER; % zaporedna številka paketa
3 f: BOOLEAN; % oznaka pravilnosti prenosa paketa
4 •: TOKEN;   % običajen žeton brez deklarirane podatkovne strukture
```

Listing 4.1: Nabor deklaracij potrebnih podatkovnih struktur.

Pri tem nam spremenljivka `p` deklarirana kot niz predstavlja podatkovno vsebino paketa, spremenljivka `n` deklarirana kot celo število števec za indeksiranje paketov, „•“ pa običajni nam že poznani žeton brez podatkovne strukture. Spremenljivka `f` predstavlja Booleanovo spremenljivko z vrednostima „`F`“ (angl. *false*) in „`T`“ (angl. *true*). Prva vrednost označuje okvaro paketa pri prenosu po omrežju, druga vrednost pa prenos paketa po omrežju brez napak.

Na osnovi predhodno deklariranih podatkovnih struktur definiramo potrebne tipe barvnih žetonov v izpisu 4.2 .

```
1 type token_1={•};
2 type token_2={n};
3 type token_3={n,f};
4 type token_4={n,p};
5 type token_5={n,p,f};
```

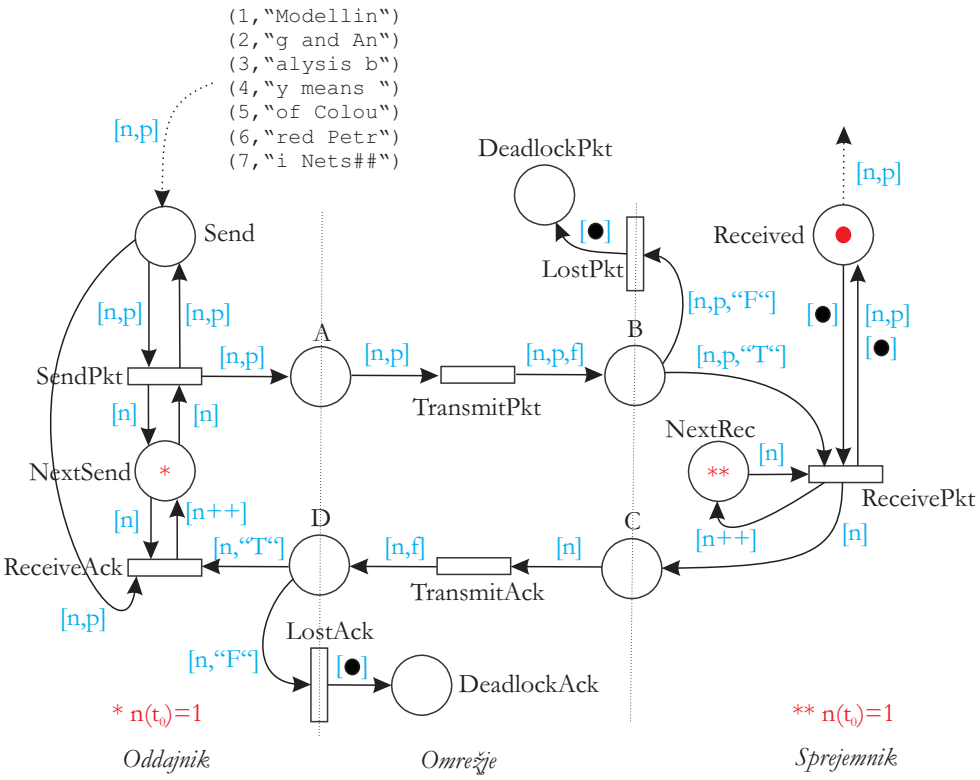
Listing 4.2: Deklaracija potrebnih različnih tipov barvnih žetonov za model enosmernega oddajno sprejemnega protokola.

Graf barvne Petrijeve mreže opisanega protokola je predstavljen na sliki 4.1 in je delno povzet po viru [17], pri čemer je na levem delu slike predstavljen oddajnik, na osrednjem neidealno omrežje, na desnem pa sprejemnik.

Na povezavah med akcijami in pogoji ter obratno s slike 4.1 so z modro barvo v oglatih oklepajih ponazorjeni tipi barvnih žetonov, ki jim je dovoljena pot po posamezni povezavi, z rdečo barvo pa začetna označitev pogojev. Opis modela protokola predstavljenega na sliki 4.1 bi bil sledeč:

- iz mrežnega nivoja v pogoj `Send` vstopajo paketi (glej zgornji levi del slike 4.1), ki jih predstavljajo barvni žetoni tipa `[n,p]`, pri čemer `n` predstavlja indeks paketa, `p` pa podatkovni niz paketa;

- akcija `SendPkt` je namenjena pošiljanju paketa preko omrežja k naslovniku pod pogojem, da je indeks paketa ustrezen; pošiljanje paketa proti omrežju v kontekstu modela predstavlja prenos barvnega žetona tipa `[n,p]` proti pogoju `A`; izvedbo akcije `SendPkt` nadzoruje pogoj `NextSend`; v njem se venomer nahaja barvni žeton tipa `[n]`, ki je na začetku inicializiran na




Slika 4.1: Graf barvne Petrijeve mreže za modeliranje enostavnega komunikacijskega protokola med sprejemnikom in oddajnikom s potrjevanjem delno povzet po viru [17]. Z modro barvo so označeni tipi barvnih žetonov, ki lahko potujejo po posameznih povezavah, z rdečo pa začetna označitev pogojev.

vrednost 1 (`n=1`); slednje pomeni, da se na začetku akcija `SendPkt` izvede le pod pogojem, da je iz pogoja `Send` možno pridobiti barvni žeton tipa `[n,p]`, ki ima inicializirano vrednost `n` na 1; v primeru, da se akcija `SendPkt` izvede, obe kopiji barvnih žetonov `[n,p]` in `[n]` iz pogojev `Send` in `NextSend` vrne v izvorna pogoja, istočasno pa barvni žeton tipa `[n,p]` odpošlje proti pogoju `A`; v primeru, da barvnega žetona z `n=1` v pogoju `Send` ni, sistem kot celota čaka na prihod takšnega barvnega žetona iz mrežnega nivoja;

- pogoj `A` predstavlja vmesnik med oddajnikom in omrežjem;
- akcija `TransmitPkt` prejme paket ali barvni žeton tipa `[n,p]`, ga razširi v barvni žeton tipa `[n,p,f]` (doda zastavico, ki označuje okvaro paketa ali njegov prenos v omrežju brez porajanja napake), ter slednjega posreduje proti pogoju `B`; dodana spremenljivka `f` predstavlja logično vrednost

tipa `Boolean`, ki lahko zavzame vrednosti „`T`“ (paket je uspešno prenešen preko omrežja) ali „`F`“ (paket je preko omrežja prispel okvarjen); inicializacija vrednosti spremenljivke `f` se izvede v notranjosti akcije `TransmitPkt`, vrednost pa se izbira naključno ali v skladu s statistikami, ki smo jih predhodno zbrali v zvezi z uspešnostjo pravilnega prenosa paketov preko omrežja;

- pogoj `B` predstavlja vmesnik med omrežjem in sprejemnikom; v primeru, da je vrednost spremenljivke `f` posameznega barvnega žetona v njem enaka „`T`“, se celotni barvni žeton tipa `[n,p,f]`, ki predstavlja paket, posreduje proti akciji `ReceivePkt`, v primeru pa da je vrednost spremenljivke `f` posameznega barvnega žetona v pogoju enaka „`F`“, se barvni žeton tipa `[n,p,f]` ali paket zavrže (preide v akcijo `LostPkt` in kasneje v pogoj `DeadlockPkt`); v tem primeru model preide v smrtni objem (angl. *deadlock*), saj oddajnik čaka na potrditev poslanega paketa, sprejemnik pa te potrditve ne bo poslal, ker paketa ni prejel;

- akcija `ReceivePkt` izvede sprejem neokvarjenega paketa ali barvnega žetona tipa `[n,p,f]`, pri čemer ima spremenljivka `f` vrednost „`T`“; izvedbo te akcije omejujeta pogoja `NextRec` in `Received`; v pogoju `NextRec` se na začetku nahaja barvni žeton tipa `[n]`, čigar vrednost je inicializirana na vrednost 1 (`n=1`); slednje pomeni, da se bo akcija `ReceivePkt` izvedla le pod pogojem, da je iz pogoja `B` možno pridobiti barvni žeton tipa `[n,p,"'T"']`, ki bo imel inicializirano vrednost `n` na 1; istočasno mora biti za izvedbo akcije `ReceivePkt` v pogoju `Received` prisoten običajni žeton „•“, ki signalizira pripravljenost na sprejem; če se akcija `ReceivePkt` izvede, se zgodi sledeče:

  - paket označen z barvnim žetonom tipa `[n,p]` in signalizator pripravljenosti sprejema „•“ se odložita v pogoj `Received`;
  - v pogoj `NextRec` se prenese nov barvni žeton tipa `[n++]`; notacija `n++` predstavlja inkrementirano vrednost spremenljivke `n`, ali indeks naslednjega paketa, ki ga pričakujemo na sprejemni strani;
  - v pogoj `C` se prenese barvni žeton tipa `[n]`, ki predstavlja potrditev prejema $n$-tega paketa;

- pogoj `C` predstavlja vmesnik med sprejemnikom in omrežjem;

- akcija `TransmitAck` prejme barvni žeton tipa `[n]`, ki predstavlja potrditveni paket, ga razširi v barvni žeton tipa `[n,f]`, ter slednjega posreduje proti pogoju `D`; dodana spremenljivka `f` s svojo vrednostjo, ki jo dodeli izvedba akcije, zopet signalizira eventuelno okvarjenost potrditvenega paketa;

- pogoj `D` predstavlja vmesnik med omrežjem in oddajnikom; v primeru, da je vrednost spremenljivke `f` barvnega žetona v njem enaka `"T"`, se barvni žeton (potrditveni paket) lahko posreduje proti akciji `ReceiveAck`,

v primeru pa da je vrednost spremenljivke `f` barvnega žetona v njem enaka `"F"`, se barvni žeton ali potrditveni paket zavrže (preide preko akcije `LostAck` v pogoj `DeadlockAck`); v tem primeru model preide v smrtni objem (angl. *deadlock*), saj oddajnik čaka na potrditev $n$-tega poslanega paketa, sprejemnik pa te potrditve ne bo več poslal, ker jo pošilja le enkrat;

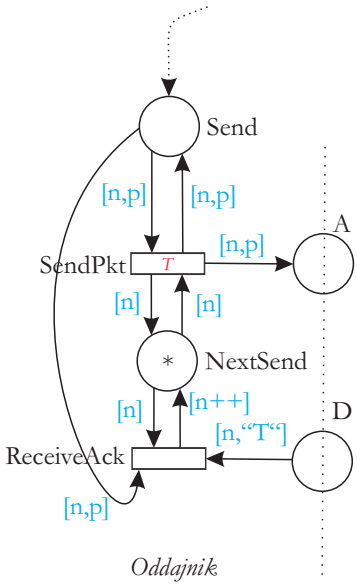
- akcija `ReceiveAck` se lahko izvede pod pogojem, da obstajajo v pogojih `NextSend`, `Send` in `D` takšni barvni žetoni ali paketi, ki se ujemajo po vrednosti spremenljivke `n`; ob izvedbi akcije `ReceiveAck` se zgodi sledeče:
  - iz pogoja `D` se odstrani barvni žeton tipa `[n,"T"]`;
  - iz pogoja `Send` se odstrani barvni žeton tipa `[n,p]`;
  - iz pogoja `NextSend` se odstrani barvni žeton tipa `[n]`;
  - v pogoj `NextSend` se naloži nov barvni žeton ali števec z inkrementirano vrednostjo spremenljivke `n`; slednji predstavlja indeks naslednjega paketa, ki naj bi šel v oddajo;

S tem smo zaključili opis dinamike v barvni Petrijevi mreži prikazani na sliki 4.1. Iz opisa postavitve modela lahko pridemo do naslednjih ugotovitev:

- akcija `SendPkt` se po prvem pošiljanju $i$-tega paketa (`n`=$i$) proži neprestano vse do prejema potrditve $i$-tega paketa, kar pripelje do poplavljanja sprejemnika z $i$-tim paketom;
- vse akcije v modelu so brez časovnega trajanja (so hipne);
- model vključuje zgolj možnosti okvar paketov, ne pa njihovega izgubljanja;
- večina pogojev sprejema samo en tip barvnega žetona, pogoj `Received` pa dva tipa in sicer običajne žetone (`"•"`) in barvne žetone tipa (`[n,p]`);
- če se v pogoju `DeadlockPkt` ali v pogoju `DeadlockAck` znajde običajni žeton (`"•"`) to pomeni, da je sistem v smrtnem objemu; izpolnjenost posameznega od obeh pogojev tako smatramo za signalizacijo obstoja smrtnega objema;

Predhodno smo že povedali, da akcija `SendPkt` ob podanem modelu s slike 4.1 poplavlja sprejemnik z $i$-tim oddajanim paketom vse do prejema njegove potrditve. Omenjeni situaciji se lahko izognemo z vpeljavo časovne periode $T$ ponovnega odpošiljanja $i$-tega paketa. Če akciji `SendPkt` dodamo trajanje $T$ urinih period, s tem zmanjšamo poplavljanje sprejemnika. Na sliki 4.2 je predstavljen izboljšan model oddajnika, pri čemer rešitev ni idealna. Oddajnik v tem primeru navkljub „hitremu“ prejemu potrditvenega paketa čaka na iztek časovne periode $T$, kar upočasnjuje oddajo novega paketa in s tem delovanje protokola kot celote.

Časovno manj potratna rešitev je prikazana na sliki 4.3, pri čemer so izboljšave prvotnega oddajnika s slike 4.1 označene z rdečo barvo. Vpeljemo nova pogoja `Wait` in `Ready` ter akcijo `ResendTimer`.



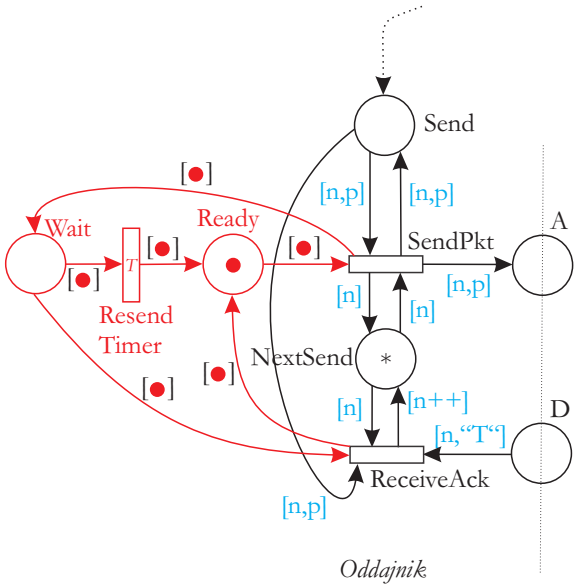

Slika 4.2: Periodično proženje akcije `SendPkt` s časovno periodo $T$, ki zmanjša poplavljanje sprejemnika. Vse akcije brez časovnega trajanja se izvedejo hipno.

Iz slike 4.3 je razvidna začetna označitev segmenta z vgrajeno periodičnostjo ponovnega pošiljanja s časovnim zamikom $T$. Ob prvi oddaji $i$-tega paketa se običajni žeton prenese iz pogoja `Ready` v pogoj `Wait`. V primeru, da pride potrditev pravočasno (pred iztekom $T$ urinih period), se omenjeni žeton iz pogoja `Wait` odstrani in preko akcije `ReceiveAck` prenese v pogoj `Ready`, s čimer so izpolnjeni pogoji za pošiljanje novega podatkovnega paketa. V primeru, da temu ni tako, žeton iz pogoja `Wait` po $T$ urinih periodah preide v pogoj `Ready` in hipno se izvede ponovno pošiljanje $i$-tega paketa.

## 4.3 Model protokola drsečega okna

V zgledu predhodnega razdelka smo ločevali med napravo, ki podatkovne pakete oddaja ter napravo, ki podatkovne pakete sprejema. V pričujočem razdelku bomo predpostavili, da imamo opravka s komunikacijo med dvema napravama, pri čemer obe podatkovne pakete lahko tako pošiljata, kot tudi sprejemata. Dvosmernost prenosa podatkovnih paketov bomo prikazali na okrnjenem zgledu modela *protokola drsečega okna* (angl. *sliding window protocol*). Omenjeni protokol deluje po sledečih principih:

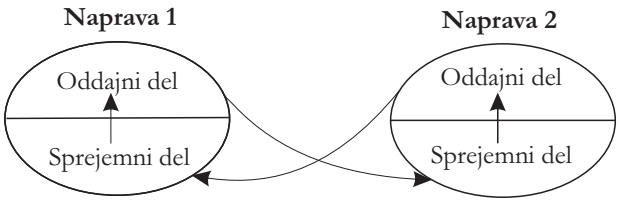
- prenosni kanal med napravama je v praksi venomer neidealen, zato protokol vsebuje potrjevanje prejetih podatkovnih paketov;




Slika 4.3: Periodično proženje akcije `SendPkt`, ki onemogoča poplavljanje sprejemnika. Vse akcije brez časovnega trajanja se izvedejo hipno.

- protokol ponuja možnost zaporednega pošiljanja večjega števila *podatkovnih paketov* proti drugi napravi, ne glede na to, da predhodno poslani podatkovni paketi še niso potrjeni; število odposlanih a nepotrjenih podatkovnih paketov je omejeno na $m$; slednje predstavlja velikost „okna“ odpošiljanja; na oddajni strani vsake naprave se vodi lista poslanih a nepotrjenih paketov, na potrditev katerih čaka naprava; lista paketov je dolžine $m$ ali krajša;

- ker sta sprejemnik in oddajnik realizirana v obeh napravah, vsaka od naprav lahko čaka na največ $m$ *potrditvenih paketov*; osnovna shema komunikacije med napravama je predstavljena na sliki 4.4;




Slika 4.4: Shema komunikacijskega kanala med dvema napravama.

- vsak prejeti potrditveni paket s strani **Naprave 2** se manifestira v odstranitvi poslanega podatkovnega paketa iz okna oddajnika **Naprave 1**; ker se s tem v oknu oddajnika **Naprave 1** sprosti prostor za en podatkovni paket, se tako omogoči oddajo enega novega podatkovnega paketa; le ta se odda takrat, ko je pripravljen za oddajo; enako pravilo velja tudi za **Napravo 2**;

Ena od tehnik, ki se v protokolih pogosta uporablja, je tehnika združevanja *podatkovnih* in *potrditvenih paketov* (angl. *piggybacking*). Osnovna ideja tehnike leži v možnosti, da potrditveni paketi opcijsko ne potujejo ločeno od podatkovnih, temveč posamezna naprava potrditev izvede na tak način, da naslednjemu podatkovnemu paketu, ki ga pošilja, doda tudi potrditev predhodno prejetega paketa. S tem dobimo optimalnejši izkoristek razpoložljive pasovne širine, saj se odposlani *kombinirani paket* skrajša za naslov prejemnika in redundantne bite samostojnega potrditvenega paketa v primerjavi s primerom, ko bi se potrditveni in podatkovni paket pošiljala ločeno.

Seveda se na tem mestu poraja vprašanje, ali bo oddaja takšnega sestavljenega ali *kombiniranega paketa* pravočasna. V primeru, da npr. **Naprava 1**, ki mora izvesti potrditev, nima pripravljene podatkovne vsebine in ta določen čas še ne bo pripravljena, mora *potrditveni paket* poslati brez podatkovne vsebine. S tem preprečimo ponovno pošiljanje *podatkovnega paketa* s strani **Naprave 2**.

V nadaljevanju bomo predstavili model opisanega protokola ponazorjen z barvno Petrijevo mrežo, za začetek pa moramo najprej deklarirati uporabljene spremenljivke, ki jih bomo uporabljali v različnih tipih barvnih žetonov. Deklaracijo podatkovnih struktur zapišemo z deklaracijskimi izrazi v izpisu 4.3.

```
1 p: CHAR[...];  % podatkovna vsebina paketa
2 n: INTEGER;    % zaporedna številka paketa
3 nACK: INTEGER; % zaporedna številka potrjevanega paketa
4 •: TOKEN;      % žeton brez podatkovne strukture
```

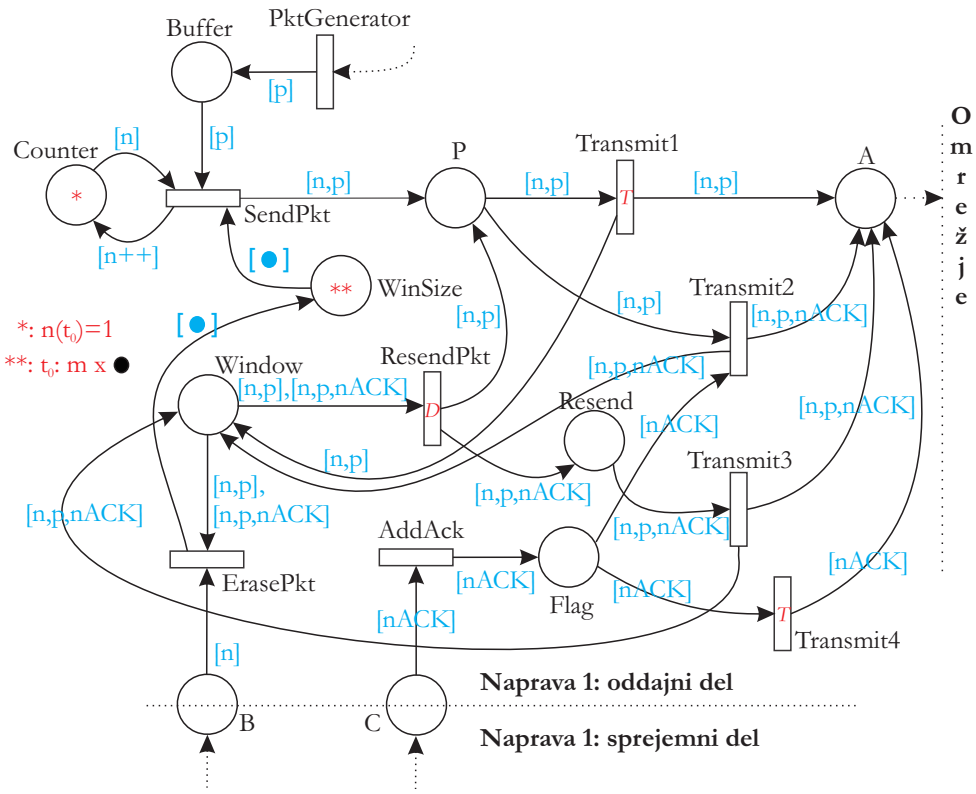
Listing 4.3: Nabor deklaracij potrebnih podatkovnih struktur.

Na osnovi predhodno deklariranih podatkovnih struktur definiramo potrebne tipe barvnih žetonov v izpisu 4.4 .

```
1 type token_1={•};      % žeton brez podatkovne strukture
2 type token_2={n};      % indeks podatkovnega paketa
3 type token_3={p};      % podatkovna vsebina paketa
4 type token_4={n,p};    % podatkovni paket
5 type token_5={n_ACK}; % potrjevalni paket (indeks potrjevanega
      paketa)
6 type token_6={n,p,nACK}; % kombinirani paket
```

Listing 4.4: Deklaracija potrebnih različnih tipov barvnih žetonov za model protokola drsečega okna.

Pri tem barvni žeton tipa `[n,p]` predstavlja podatkovni paket, barvni žeton tipa `[n_ACK]` potrjevalni paket, barvni žeton tipa `[n,p,nACK]` pa kombinirani paket. Pri postavitvi modela se bomo zaradi preglednosti omejili zgolj




Slika 4.5: Model oddajnega dela protokola drsečega okna ponazorjen z barvno Petrijevo mrežo.

na oddajni del ene od obeh naprav. Model oddajnega dela prve naprave je predstavljen na sliki 4.5 in je identičen modelu oddajnega dela druge naprave.

Ključne točke modela s slike 4.5 so sledeče:

- pogoj `A` predstavlja vmesnik med oddajnikom in omrežjem; vanj se stekajo različne vrste paketov, ki jih ponazarjajo barvni žetoni tipov `[n,p]`, `[n,p,nACK]` in `[nACK]`; omrežje iz pogoja `A` (vmesnika) samodejno prevzema pakete;
- pogoj `B` predstavlja vmesnik, preko katerega sprejemni del naprave prenese oddajnemu delu indeks potrjenega paketa preko barvnega žetona tipa `[n]`; oddajni del se bo na ta barvni nabor odzval z brisanjem $n$-tega paketa iz okna (pogoja `Window`) na osnovi akcije `ErasePkt` in zviševanjem števca prostih mest v oknu (prenos običajnega žetona v pogoj `WinSize`);
- pogoj `C` predstavlja vmesnik, preko katerega sprejemni del naprave prenese oddajnemu delu indeks sprejetega podatkovnega paketa preko barvnega

žetona tipa `[nACK]`; omenjeni žeton predstavlja potrditveni paket, ki ga mora oddajnik odposlati v vmesnik `A`, bodisi kot samostojen potrditveni paket, bodisi kot kombinirani paket; preko akcije `AddAck` bo indeks potrditvenega dela paketa posredovan v pogoj `Flag`, kjer čaka na odpremo;

- pošiljanje različnih vrst paketov proti omrežju (proti pogoju `A`) poteka iz sledečih akcij:
  - akcija `Transmit1` je zmožna prve in ponovne oddaje podatkovnega paketa (barvnega žetona tipa `[n,p]`) proti pogoju `A` (omrežju) v primeru odsotnosti barvnega žetona tipa `[nACK]` (potrditvenega paketa) v pogoju `Flag`; z nekim infinitezimalnim časom trajanja $T$ akcija `Transmit1` daje prednost izvajanju akcije `Transmit2` v primeru prisotnosti potrditvenega paketa (v pogoju `Flag` se nahaja barvni žeton tipa `[nACK]`); ob izvedbi akcije `Transmit1` se kopija poslanega žetona tipa `[n,p]` (poslanega paketa) prenese tudi v pogoj `Window`, ki vrši funkcijo drsečega okna;
  - akcija `Transmit2` je konkurenčna akciji `Transmit1` in posreduje kombinirani paket (barvni žeton tipa `[n,p,nACK]`) proti pogoju `A`; izvede se ob prisotnosti barvnega žetona tipa `[n,p]` (podatkovnega paketa) v pogoju `P` in ob prisotnosti barvnega žetona tipa `[nACK]` (potrditvenega paketa) v pogoju `Flag`; akcija sestavi in nato odda kombinirani paket - barvni žeton tipa `[n,p,nACK]`; ob izvedbi akcije `Transmit2` se kopija poslanega žetona tipa `[n,p,nACK]` prenese tudi v pogoj `Window`, ki vrši funkcijo drsečega okna;
  - akcija `Transmit3` je zmožna ponovne oddaje kombiniranega paketa (barvnega žetona tipa `[n,p,nACK]`) proti omrežju, ki ga prevzame iz pogoja `Resend`; kombiniranemu paketu ne dodaja novih potrditev (barvnih žetonov tipa `[nACK]`), ki eventuelno čakajo v pogoju `Flag`; ob izvedbi akcije `Transmit3` se kopija poslanega žetona tipa `[n,p,nACK]` odloži v pogoj `Window`, ki vrši funkcijo drsečega okna;
  - akcija `Transmit4` proti omrežju pošilja zgolj samostojne potrditvene pakete (barvne žetone tipa `[nACK]`) pod pogojem, da v pogoju `P` ni pripravljenih podatkovnih paketov; opredeljena je z infinitezimalnim časovnim trajanjem $T$, kar pomeni, da barvni žeton tipa `[nACK]` čaka v pogoju `Flag` $T$ urinih period; če ga v tem obdobju ne prevzame akcija `Transmit2` z namenom združitve s podatkovnim paketom (barvnim žetonom tipa `[n,p]`), se potrditveni paket (barvni žeton tipa `[nACK]`) odpošlje samostojno proti pogoju `A`; potrditvenega paketa se s proženjem akcije `Transmit4` ne odlaga v okno (pogoj `Window`);

Opis preostalih delov modela s slike 4.5 je naveden v sledečih alineah:

- akcija `PktGenerator` generira podatkovne pakete ali barvne žetone tipa `[p]`; pri tem `p` predstavlja podatkovno vsebino podatkovnega paketa; akcijo sproži pulz iz zunanjega sveta (npr. iz mrežnega nivoja);

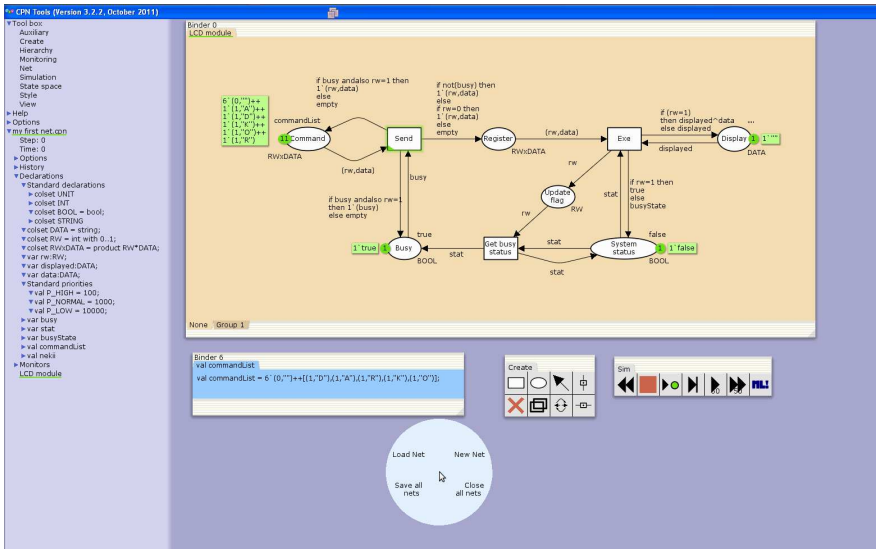
- v pogoju `Buffer` čakajo podatkovni paketi (barvni žetoni tipa `[p]`) po FIFO principu na oddajo;
- pogoj `Counter` dodeljuje indeks podatkovnemu paketu (barvnemu žetonu tipa `[p]`); na začetku simulacije dinamike modela je vrednost spremenljivke `n` barvnega žetona tipa `[n]` nastavljena na 1 ($n(t_0) = 1$), z vsako oddajo paketa (aktivacijo akcije `SendPkt`) pa se vrednost spremenljivke `n` inkrementira ($n = n + 1$);
- akcija `SendPkt` sestavi barvna žetona tipov `[n]` in `[p]` v enoten barvni žeton tipa `[n,p]` ter takšen žeton (indeksiran podatkovni paket) odpošlje v naslednjo fazo oddaje podatkovnega paketa; akcija se lahko izvede samo pod pogojem, da „okno“ še ni polno, ali povedano drugače, ko je v pogoju `WinSize` na razpolago še kak običajen žeton;
- pogoj `WinSize` vsebuje števec prostih mest v „oknu“; vrednost števca ponazarja število običajnih žetonov v tem pogoju; začetno število žetonov v tem pogoju tako predstavlja velikost „okna“; na začetku je le ta inicializirana na število $m$, kar pomeni, da je v pogoju na začetku $m$ običajnih žetonov; iz začetne označitve pogoja je razvidno, da preko akcije `SendPkt` lahko proti omrežju odide $m$ barvnih žetonov (podatkovnih paketov), v nadaljevanju pa je proženje akcije `SendPkt` odvisno od vračanja običajnih žetonov v pogoj `WinSize` s strani akcije `ErasePkt` (ali prejema potrditev predhodno poslanih paketov);
- v pogoj `P` se stekajo barvni žetoni tipa `[n,p]` ali podatkovni paketi, ki so bodisi novi ali predhodno že poslani; v primeru, da zastavica `Flag` ni dvignjena, se odpošiljajo proti omrežju preko akcije `Transmit1` kot podatkovni paketi, v primeru pa da je zastavica `Flag` dvignjena (v čakanju je tudi potrjevalni paket), se odpošiljajo proti omrežju preko akcije `Transmit2` kot kombinirani paketi;
- pogoj `Window` vrši funkcijo drsečega okna ali hrambe predhodno že poslanih paketov, na katerih potrditev čakamo; v njem je lahko največ $m$ barvnih žetonov, ki so lahko različnih tipov;
- akcija `ResendPkt` zagotavlja ponovno pošiljanje podatkovnega ali kombiniranega paketa v primeru, da nismo prejeli potrditve predhodno poslanega paketa v $D$ urinih periodah;

Obsežnejši zgled modela protokola drsečega okna (angl. *sliding window*) temelječ na barvnih Petrijevih mrežah je predstavljen v delu [19].

## 4.4 Programsko orodje CPN Tools za delo z barvnimi Petrijevimi mrežami

Za lažje delo z barvnimi Petrijevimi mrežami je bilo v preteklosti razvitih kar nekaj programskih orodij. Eno od popularnejših nekomercialnih in prosto dose-

gljivih je orodje *CPN Tools*[1]. Na sliki 4.6 je predstavljen uporabniški vmesnik z naloženim modelom. Orodje omogoča osnovne funkcije gradnje modela (dodajanje pogojev, akcij, povezav, itd.), modularnost izgradnje modela, upravljanje s podatkovnimi tipi barvnih žetonov in samodejno preverja sintaksno pravilnost modela. Posamezni model se shrani v datoteko s končnico *.cpn*, ki temelji na XML datotečni strukturi.




Slika 4.6: Uporabniški vmesnik orodja *CPN Tools* [19].

Orodje ponuja tudi izvajanje simulacij na osnovi zasnovanega modela. Slednje se izvajajo grafično. Tako lahko skozi čas simulacije vizuelno spremljamo število žetonov v posameznih pogojih. Osnovne funkcije za izvajanje simulacij delimo na *interaktivne* in *samodejne*. Pod interaktivnimi imamo v mislih tiste, kjer se del simulacijske decizije prepušča uporabniku (npr. ročno izvajanje akcij, ročno izbiranje konkurenčnih ali splošneje omogočenih akcij), pod samodejnimi pa samodejno (avtomatizirano) izvajanje vključno z reševanjem nedeterministične izbire vejanj, reševanjem konkurenčnosti ter izvajanjem simulacij za vnaprej predvideno število simulacijskih korakov ali realen čas, ali do izpolnitve vnaprej podanih pogojev (npr. konkretne označitve).

Pri proženju samodejne simulacije brez definicije konca, ki bo tekla vse do dosega končne označitve, je pomembno, da se zavedamo, da ni nujno, da se bo simulacijski tok ustavil. Do tega lahko pride bodisi, ker v sistemu ni končne označitve, ali pa zaradi tega, ker se je sistem vsled nedeterminističnim vejanjem slednji izognil. Tovrstni dogodek nas že opozarja, da obstaja možnost preslabe definicije zasnove rešitve problema (modela).

[1]Orodje CPN Tools je prosto dosegljivo na spletnem naslovu http://cpntools.org/

Orodje za uspešnejše razhroščevanje omogoča vpeljavo *nadzornikov*, ki imajo vnaprej predvidene nadzorne funkcije. Slednje definirajo pogoje pod katerimi pride do začasne (k interakciji je povabljen uporabnik), ali dokončne ustavitve simulacije. Podpira tudi analizo prostora dosegljivih stanj. Rezultate analize orodje predstavi z usmerjenim grafom, v katerem kot vozlišča nastopajo označitve (stanja sistema). Slednje smo v poglavju o Petrijevih mrežah predstavili z *drevesom dosegljivih stanj*. Analiza prostora stanj nam poleg drevesa dosegljivih stanj predstavi še naslednje pomembne značilnosti modeliranega sistema:

- velikost prostora stanj;
- obstoj $k$-omejenosti v posameznih pogojih glede na različne vrste tipov barvnih žetonov;
- obstoj domače označitve, ki je dosegljiva iz vseh preostalih označitev (dobrodošla značilnost, ki ponazarja, da se je sistem po opravljeni nalogi sposoben vrniti v neko začetno stanje);
- pogostosti proženja posameznih akcij, itd.;

Dodatne primere modelov barvnih Petrijevih mrež s področja računalniških komunikacij najdemo v delu [20].

# Literatura


[1] N. C. Hock, *Queuing Modelling Fundamentals.* John Wiley & Sons, Chichester, Anglija, 1996.

[2] M. Anu, "Introduction to modeling and simulation," in *Proceedings of the 29th conference on Winter simulation* (S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson, eds.), pp. 7–13, 1997.

[3] L. Kleinrock and R. Gail, *Queuing systems, problems and solutions.* John Wiley & Sons, New York, ZDA, 1996.

[4] N. Zimic and M. Mraz, *Temelji zmogljivosti računalniških sistemov.* Založba FE in FRI, Ljubljana, Slovenija, 2006.

[5] R. Jamnik, *Verjetnostni račun in statistika.* Društvo matematikov, fizikov in astronomov socialistične republike Slovenije, Zveza organizacij za tehnično kulturo Slovenije, Ljubljana, Slovenija, 1986.

[6] K. S. Trivedi, *Probability and Statistics with Reliability, Queueing and Computer Science Applications.* John Wiley & Sons Inc., New York, ZDA, 2002.

[7] H. Stöcker, *Matematični priročnik z osnovami računalništva.* Tehnična založba Slovenije, Ljubljana, Slovenija, 2006.

[8] J. Virant, *Modeliranje in simuliranje računalniških sistemov.* Didakta, Radovljica, Slovenija, 1991.

[9] J. F. Shortle, J. M. Thompson, D. Gross, and C. M. Harris, *Fundamentals of queueing theory.* John Wiley & Sons, Hoboken, ZDA, 2018.

[10] J. L. Peterson, *Petri Net Theory and the Modeling of Systems.* Prentice Hall Inc., Englewood Cliffs, ZDA, 1981.

[11] J. Bordon, M. Moškon, N. Zimic, and M. Mraz, "Semi-quantitative Modelling of Gene Regulatory Processes with Unknown Parameter Values Using Fuzzy Logic and Petri Nets," *Fundamenta Informaticae*, vol. 160, no. 1–2, pp. 81–100, 2018.

[12] J. Virant, *Logične osnove odločanja in pomnjenja v računalniških sistemih.* Založba FE in FRI, Ljubljana, Slovenija, 1996.

[13] T. Murata, "Petri Nets: Properties, Analysis and Applications," *Proceedings of The IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[14] W. G. Schneeweiss, *Petri Nets for Reliability Modeling.* LiLoLe Verlag, 1999.

[15] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks.* Prentice Hall Inc., Boston, ZDA, 2011.

[16] N. Jensen and L. Kristensen, *Coloured Petri Nets.* Springer,, 1998.

[17] K. Jensen, "A Brief Introduction to Coloured Petri Nets," in *Proceedings of the Third International Workshop on Tools and Algorithms for Construction and Analysis of Systems (TACAS '97)*, 1997.

[18] K. Jensen, *Coloured Petri Nets Basic concepts.* Springer, 1997.

[19] D. Božić, *Analiza in zgled uporabe programskega orodja CPNTools za postavljanje modelov dinamičnih sistemov.* Diplomsko delo FRI-UL, 2012.

[20] M. Dolenc, *Verifikacija komunikacijskih protokolov na osnovi barvnih Petrijevih mrež.* Diplomsko delo FRI-UL, 2015.