Slovenská technická univerzita

Fakulta informatiky a informačných technológií
Ilkovičova 3, 812 19 Bratislava

Diagnostika porúch diskrétnych udalostných systémov založená na modeloch a aplikácie v informatike

Tímový projekt

Tím č. 5
Bc. Tomáš Bartek
Bc. Jozef Ferenčík
Bc. Lukáš Humený
Bc. Peter Kubanda
Bc. Pavol Kubík
Bc. Michal Kurtý

Vedúci tímového projektu: Ing. Jana Flochová, PhD.

Zadanie

Diagnostika porúch diskrétnych udalostných systémov založená na modeloch a aplikácie v informatike.

V projekte ide o analýzu, programovanie a overovanie metód diagnostiky porúch diskrétnych udalostných systémov založenej na modeloch systému (konečný automat, stochastický automat, Petriho sieť atď.) a o navrhnutie a testovanie príkladov z oblasti informatiky, najmä počítačových sieti, architektúry softvéru, hardvéru a podobne.

Softvérový nástroj na analýzu a diagnostiku možnosti poruchy si študenti môžu jednak zvoliť (na Internete je mnoho voľne dostupného materiálu najmä na modelovanie diskrétnych systémov) alebo si ho môžu naprogramovať v C++, Visual C++, C#, Java, pripadne môžu prepísať niektoré modely i do VHDL.

Zoznam použitých skratiek

PN (Petri Nets) – Petriho siete
SPN (Stochastic Petri Nets) – Stochastické Petriho siete
PPN (Probability Petri Nets) – Pravdepodobnostné Petriho siete
P (Places) – Miesta
LAN (Local Area Network) – Lokálna sieť
T (Transitions) - Prechody
F (Fire) – Spustenie prechodu
M0 (Initial marking) – Počiatočné značkovanie
CPN (Colored Petri net) – Farebné Petriho siete

1 Obsah

1 OBSAH 3
2 ÚVOD 5
2.1 PREHĽAD DOKUMENTU 5
3 ZÁKLADY MODELOVANIA A SIMULOVANIA SYSTÉMOV 6
3.1 KLASIFIKÁCIA SYSTÉMOV 6
3.1.1 DETERMINISTICKÉ A NEDETERMINISTICKÉ (STOCHASTICKÉ) SYSTÉMY 6
3.1.2 SPOJITÉ A DISKRÉTNE SYSTÉMY 7
3.1.3 CENTRALIZOVANÉ A DISTRIBUOVANÉ SYSTÉMY 7
3.2 MODELY SYSTÉMOV 8
4 PRINCÍPY MODELOVANIA A SIMULOVANIA SYSTÉMOV 9
4.1 KLASIFIKÁCIA MODELOV 10
4.2 SPÔSOBY POPISU SIEŤOVÝCH SYSTÉMOV 11
4.2.1 KONEČNÝ AUTOMAT 11
4.2.2 DES 11
4.2.3 PETRIHO SIETE 12
5 PETRIHO SIETE 13
5.1 TYPY PETRIHO SIETÍ 14
5.1.1 C/E PETRIHO SIETE 15
5.1.2 P/T PETRIHO SIETE 16
5.1.3 DETERMINISTICKÉ ČASOVANÉ PETRIHO SIETE 17
5.1.4 STOCHASTICKÉ ČASOVANÉ PETRIHO SIETE 17
5.1.5 FAREBNÉ PETRIHO SIETE (CPN) 18
5.2 VLASTNOSTI PETRIHO SIETI 20
5.2.1 VLASTNOSTI SPRÁVANIA SA PS 20
5.2.2 VLASTNOSTI ŠTRUKTÚRY PS 20
5.2.3 DOSIAHNUTEĽNOSŤ ZNAČKOVANIA A POKRYTIE 21
5.2.4 OHRANIČENOSŤ A BEZPEČNOSŤ 22
5.2.5 ŽIVOSŤ A REVERZIBILNOSŤ 22
5.2.6 BEZKONFLIKTNOSŤ 23
5.2.7 KONZERVATÍVNOSŤ 24
6 PROBLEMATIKA LOKALIZÁCIE PORÚCH V KOMUNIKAČNÝCH SYSTÉMOCH 25

6.1 DETEKCIA A IZOLÁCIA PORÚCH 25
6.2 TECHNIKY PRECHÁDZANIA MODELU 26
6.3 GRAFOVO TEORETICKÉ TECHNIKY 26
7 MODELOVANIE PETRIHO SIETÍ 27
7.1 PROGRAMOVÉ NÁSTROJE 28
7.1.1 CPN TOOLS (COLOURED PETRI NETS) 28
7.1.2 PLATFORM INDEPENDENT PETRI NET EDITOR 2 (PIPE2) 29
7.1.3 NETLAB 30
7.1.4 HPSIM 31
7.1.5 ZHODNOTENIE MODELOVACÍCH NÁSTROJOV 32
8 PROTOKOL OSPF 34
8.1 ZÁKLADNÁ FUNCIONALITA OSPF SMEROVAČOV 34
8.2 TYPY SIETÍ V OSPF 35
8.3 TYPY OSPF PAKETOV 35
8.4 TYPY SPRÁV LINK STATE ADVERTISEMENTS 36
8.5 VYTVÁRANIE SUSEDSTIEV MEDZI OSPF SMEROVAČMI 37
9 ŠPECIFIKÁCIA A NÁVRH RIEŠENIA 39
9.1 SIMULÁCIA VYBRANÝCH MECHANIZMOV PROTOKOLU OSPF 40
9.1.1 ČO CHCEME SIMULOVAŤ 40
9.1.2 AKO CHCEME SIMULOVAŤ? 40
9.2 KONCEPCIA MODELU 41
9.3 PROTOTYP MODELU 43
9.3.1 PRECHOD ZO STAVU DOWN DO STAVU INIT 45
9.3.2 PRECHOD ZO STAVU INIT DO STAVU 2WAY 45
9.3.3 PRECHOD ZO STAVU 2 WAY DO STAVU EXSTART 46
9.3.4 PRECHOD ZO STAVU EXSTART DO STAVU EXCHANGE 46
9.3.5 PRECHOD ZO STAVU EXCHANGE DO STAVU LOADING/FULL 47
10 PLÁN ĎALŠEJ PRÁCE 48
11 POUŽITÁ LITERATÚRA 49

2 Úvod

Cieľom tohto tímového projektu je analyzovať a diagnostikovať možné poruchy v diskrétnych udalostných systémoch pomocou modelov Petriho sietí. Pre správne navrhnutie modelu nejakého systému je dobre o ňom mať čo najviac informácií. Je potrebné poznať jeho správanie v jednotlivých situáciách, potreby na riadenie, fyzikálne či iné obmedzenia daného procesu. Ďalším krokom je presne si zadefinovať aké sú požiadavky na daný systém, čo chceme aby systém vykonával. Dôležité je určiť si správne priority, aby nedochádzalo ku kolíziám. Podľa týchto požiadaviek sa pokúsime v tejto práci vytvoriť čo najlepší všeobecný model diskrétneho systému. Medzi najpoužívanejšie spôsoby vytvorenia modelov patria Konečné automaty a Petriho siete. Podrobnejšie sa v tejto práci budeme venovať Petriho sieťam, keďže pre tento spôsob modelovania sme sa rozhodli.

2.1 Prehľad dokumentu

V úvodných kapitolách analytickej časti dokumentu (kapitola 3 a 4) približujeme čitateľovi základné princípy a rozdelenia modelovania systémov, vysvetľujeme základné pojmy súvisiace s oblasťou modelovania a simulácie.

V ďalšej kapitole (kapitola 5) opisujeme problematiku Petriho sietí, ich základnú teóriu, rozdelenie a vlastnosti.

V kapitole č.6 stručne popisujeme princípy a techniky analýzy a lokalizácie porúch v simulovaných systémoch.

Kapitola č.7 približuje čitateľovi a porovnáva najrozšírenejšie programové prostriedky, pomocou ktorých prebieha simulácia a analýza navrhnutých modelov.

V kapitole č.8 je popisaná špecifikácia navrhovaného systému a funkcionalita vybraného simulovaného sieťového protokolu - OSPF.

3 Základy modelovania a simulovania systémov

Modelovanie systému je činnosť, ktorá človeku umožňuje uvažovať o reálnom svete a na základe získaných poznatkov ho cieľavedome ovplyvňovať. Pod pojmom systém obvykle rozumieme abstrakciu reality, pričom sa zameriavame len na tie skutočnosti, ktoré sú relevantné pre naše skúmanie [1]. Keď hovoríme o modelovaní systému, znamená to, že ho reprezentujeme v nejakom inom prostredí (napríklad v jazyku alebo forme, ktorá je zrozumiteľná počítaču), kde je nevyhnutné uvažovať jeho vnútorné charakteristiky, ako je vzájomná interakcia medzi systémom a jeho okolím.

Z matematického hľadiska je to množina prvkov spojených navzájom s okolím za pomoci interakcií, čím sa vytvárajú vlastnosti umožňujúce plnenie účelových a cieľových funkcií. Štruktúrov sa rozumie vnútorné usporiadanie systému vyjadrené vzájomnými väzbami a pôsobením zložiek vnútornej organizácie celku, alebo jeho častí. Formálny popis systému je možné vyjadriť tvarom S(X,R), kde X je množina prvkov a R je množina relácií alebo vzťahov medzi prvkami. Najjednoduchší systém je zostavený z dvojice prvkov, v tomto prípade množina R obsahuje iba jedinú reláciu – vzťah medzi dvoma prvkami. Všeobecne sa dá povedať, že každý prvok systému je sám systémom a označuje sa ako podsystém.

3.1 Klasifikácia systémov

V teórii systémov existuje viacero spôsobov, podľa ktorých je možné klasifikovať systémy, medzi najčastejšie používané patria delenia opísané v nasledujúcich kapitolách.

3.1.1 Deterministické a nedeterministické (stochastické) systémy

Hovoríme, že systém je deterministický, ak môžeme určiť jeho výstupné veličiny a stav v ľubovoľnom okamihu $t > t_o$ s istotou. To znamená, že jeho správanie je jednoznačne určené jeho stavom a podnetmi.

V nedeterministickom systéme sa výstupy nereprodukujú. Jeho výstupné veličiny a stav možno určiť len s určitou pravdepodobnosťou alebo inými štatistickými metódami. Medzi nedeterministické systémy patria tzv. systémy stochastické, teda systémy, v ktorých prebiehajú stochastické procesy ako väzby medzi jeho prvkami, pričom pojem stochastický proces definujeme ako zmenu systému v čase podmienenou vplyvom nekontrolovateľných náhodných faktorov.

3.1.2 Spojité a diskrétne systémy

Systém je označovaný ako spojitý, ak vstupné, výstupné a stavové veličiny sú definované pre každé t v určitom intervale $t_0 \leq t \leq t_k$ a správanie je opísané diferenciálnou, algebraickou, diferenčnou rovnicou alebo systémom rovníc, z čoho vyplýva, že dej prebieha vo všetkých prvkoch súčasne.

Systém je diskrétny, ak vstupné, výstupné a stavové veličiny sú definované len v diskrétnych časových okamihoch t_n, obvykle $t = n.T$, kde n je postupnosť spravidla celých čísel z intervalu $t_0 \leq tn \leq t_k$. U diskrétnych systémov pozorujeme len skokové zmeny stavu, spôsobené výskytmi udalostí [2]. Medzi vlastnosti diskrétnych systémov patrí najmä stabilita, možnosť riadenia a pozorovateľnosť, vďaka čomu sú najčastejšie používaným typom pri modelovaní väčšiny typov zložitých dynamických systémov, ako sú počítačové systémy, distribuovanej či paralelnej povahy, workflow systémy, sieťové systémy či komunikačné protokoly.

V súčasnosti sú v teórii systémov najrozšírenejšími objektmi skúmania a modelovania tzv. systémy diskrétnych udalostí. V tomto prípade je každý objekt reprezentovaný ako jednotka definovaná množinou premenných, kde ich hodnota v istom časovom úseku opisuje stav systému. Tak ako sa jednotlivé jednotky v čase menia, generujú tzv. udalosti, ktoré sú schopné modifikovať stav systému.

V teórii modelovania existujú premenné, ktoré môžu nadobudnúť nekonečnú množinu hodnôt a nazývajú sa spojité premenné. Ostatné premenné, tzv. diskrétne premenné, môžu nadobudnúť hodnoty len z jednej konečnej množiny hodnôt. Príkladom spojitej premennej je čas príchodu a odchodu paketu v komunikačnej sieti, príkladom diskrétnej premennej je počet uzlov, ktorými paket prejde. Podľa typu premenných použitých pre opis systému, hovoríme o modeli so spojitými stavmi, ak sú jeho premenné spojitej povahy, a v prípade, že použité premenné sú diskrétne, hovoríme o modeli s diskrétnymi stavmi. Prepojenie týchto dvoch typov modelov predstavuje tzv. zmiešaný alebo hybridný model.

3.1.3 Centralizované a distribuované systémy

Distribuované systémy sú systémy, v ktorých sa jednotlivé časti procesov vykonávajú simultánne na dvoch alebo viacerých počítačoch, ktoré spolu komunikujú cez počítačovú sieť, zatiaľ čo centralizovaný systém spracúva procesy na jedinom počítači. Distribuované systémy sú špecifickým typom paralelných systémov.

3.2 Modely systémov

Pod pojmom model vo všeobecnosti rozumieme vytvorený systém, ktorý je určitým (niekedy i podstatným spôsobom) zjednodušením originálu modelovaného systému. Medzi originálom a jeho modelom existuje homomorfný vzťah zobrazenia, pričom rozlišujeme medzi modelmi abstraktnými (teda myšlienkovými, teoretickými), nad ktorými môžeme viesť logické úvahy a modelmi simulačnými (konkrétnymi, fyzicky realizovanými, spustiteľnými), na ktorých môžeme vykonávať simulačné experimenty.

Niektoré modely realizované prostredníctvom počítačov môžu patriť do oboch spomínaných kategórií súčasne, nakoľko poskytujú teóriu, umožňujúcu logickým odvodzovaním dokazovať vlastnosti modelu a zároveň je možné s týmto modelom vykonávať simulačné experimenty. Zatiaľ čo logické odvodzovanie vlastností umožňujú formálne modely založené na nejakom jednoduchom matematicky dobre spracovateľnom formalizme, počítačovú simuláciu umožňujú všetky vykonateľné modely, ktoré sú zapísané v nejakom programovacom jazyku.

Správanie dynamického systému môžeme charakterizovať stavovou premennou a zmenami jej hodnoty v čase, pričom mapovanie času na stav (priebeh hodnôt stavu v čase) nazývame procesom.

4 Princípy modelovania a simulovania systémov

Vytváranie modelov takých systémov, ktoré sú predmetom nášho skúmania, je veľmi rozšírenou činnosťou človeka už od počiatkov jeho existencie. Obecne sa dá povedať, že naša predstava sveta je vlastne modelom reality, teda okolitého sveta, ktorý môžeme skúmať. Pri modelovaní vychádzame z informácií o systéme, ktoré sú dostupné. Model, ktorý vznikne, reprezentuje formalizované znalosti o modelovanom systéme z hľadiska, ktoré chceme skúmať a obvykle pokrýva len tú časť popisu celého systému, ktorá je pre daný účel podstatná. Nakoľko model vždy vychádza z nejakej podmnožiny našich znalostí, ktoré sú neúplné, môžeme modelovať výlučne to, čo sme schopní pochopiť a opísať. Simulačné modelovanie teda predstavuje proces transformácie znalostí zo strojovo neakceptovateľnej reprezentácie na reprezentáciu akceptovateľnú počítačom.

Základnými etapami modelovania a simulácie sú [3]:

1. Vytvorenie abstraktného modelu – formovanie zjednodušenia popisu skúmaného systému
2. Vytvorenie simulačného modelu – zápis abstraktného modelu formou programu
3. Simulácia – experimentovanie s prezentáciou simulačného modelu
4. Analýza a interpretácia výsledkov - overenie správnosti modelu

Počítačový model je napodobenina systému iným systémom, teda počítačovým programom. Model systému musí napodobňovať všetky pre naše účely podstatné vlastnosti systému, nakoľko reprezentuje formalizované znalosti o modelovanom systéme z hľadiska, ktoré je predmetom skúmania.

Modelovanie je proces vytvárania modelu systému na základe určitých znalostí. Tento proces je obecne veľmi náročný a často vyžaduje znalosti z viacerých oborov. Kvalita vytvoreného modelu často zásadným spôsobom ovplyvňuje výsledky získané neskorším experimentovaním.

4.1 Klasifikácia modelov

Presné rozdelenie modelov do kategórií je pomerne náročné, pretože často neexistuje jednoznačné klasifikačné kritérium. Modely môžu byť delené napríklad podľa spôsobu ich matematického popisu, podľa úrovne abstrakcie, podľa metód implementácie (paralelné, distribuované) a podľa celej rady ďalších kritérií. V rámci potrieb tejto práce sa obmedzíme na klasifikačné kritérium podľa návrhu modelov:

- Konceptuálne modely – sú to neformálne popisy systémov, ktoré definujú základnú štruktúru systému (objekty a ich vzťahy), nedefinujú však kategórie bežné v teórii sieťových systémov (stav, udalosť, funkcia). Obvykle sa používajú v počiatočnej fáze modelovania a majú formu textov alebo obrázkov.

- Deklaratívne modely – modely, ktoré slúžia na popis prechodov medzi stavmi systému, pričom samotný model je definovaný stavmi a udalosťami, ktoré spôsobujú prechod z jedného stavu do druhého za istých podmienok. Vhodné sú predovšetkým pre diskrétne modely a obvykle bývajú zapuzdrené do objektov, ktoré vytvárajú charakteristickú hierarchickú štruktúru. Patria sem konečné automaty (deterministické i nedeterministické - Markovove modely) či Petriho siete.

- Funkcionálne modely – modely zobrazené vo forme grafov, ktoré zobrazujú funkcie a premenné vo forme uzlov grafu. Medzi predstaviteľov tohto typu modelov patria systémy hromadnej obsluhy s frontami, kompartmentové systémy či grafy signálových tokov.

- Modely popísané rovnicami – modely, ktoré sú charakteristické svojim popisom rovnicami algebraickými, diferenciálnymi či diferenčnými alebo neorientovanými grafmi (elektrické schémy a tzv. Bond Graphs).

- Priestorové modely – tento typ modelov rozdeľuje systém na priestorovo menšie ohraničené podsystémy, ktoré majú definované správanie, cez ktoré dokážeme pochopiť komplexné fungovanie systému. Medzi ne patria tzv. celulárne automaty či L-systémy.

- Multimodely – medzi tento typ modelov patria modely, ktoré sú zložené z iných modelov a sú obvykle heterogénnej povahy, teda popísané rôznymi spôsobmi. Patria sem kombinované modely (prepojenie spojitých a diskrétnych

charakteristík), modely s neurčitosťou (fuzzy modely) a prepojené simulačné systémy (HLA).

4.2 Spôsoby popisu sieťových systémov

Medzi základné spôsoby popisu sieťových systémov patria následovné formy:

4.2.1 Konečný automat

Konečný automat je zúžením všeobecného prechodového systému. Je to abstraktný model slúžiaci na modelovanie systémov, u ktorých je možné vymedziť konečný počet stavov a konečný počet vonkajších podnetov. Stav tohto systému sa zmení iba na základe vonkajšieho podnetu a to jednoznačne. Množiny vstupov, výstupov, stavov sú konečné. Od toho je odvodený pojem konečný automat. [2]

4.2.2 DES

DES (Diskrétny udalostný systém) je charakterizovaný množinou stavov *S*, množinou asynchrónnych udalostí v čase *E*, ktoré zodpovedajú jednotlivým prechodom medzi stavmi a prenosovou funkciou:

$$D\ (D\colon S \times E \rightarrow S \cup \Lambda$$

kde Λ indikuje nulový element, použitý na identifikáciu nedefinovaného prechodu).

Potom možno DES opísať nasledovne:

$$\sum = (S,E,D).$$

V prípade, že je známa informácia o výskyte *k*-tej udalosti e_k v case t_k ($\{e_k,t_k\}_{k=0,1,\ldots}$), hovoríme o tzv. časových diskrétnych udalostných systémoch. V takom prípade možno diskrétny udalostný systém charakterizovať ako množinu:

$$\sum = (S,E,D,F)$$

kde

$$F = \{F_e(.)\colon e \in E\}$$

je množina pravdepodobnosti distribučnej funkcie asociovanej typom udalosti. Náhodná premenná τ_e charakterizovaná *Fe*(.) je označovaná ako životnosť udalosti *(angl. lifetime*) a následne aj množina *F* je označená ako generátor životnosti udalostí daného DES. [4]

4.2.3 Petriho siete

Petriho sieť je matematická reprezentácia diskrétnych distribuovaných systémov. Graficky reprezentuje štruktúru distribuovaného systému ako orientovaný bipartitný graf s ohodnotením. Petriho siete sú nástrojom pre modelovanie a simulovanie diskrétnych javov. Je to grafová štruktúra, ktorá obsahuje dva druhy uzlov - miesta a prechody, d'alej obsahuje hrany a značky, ktoré sa vyskytujú v miestach. Simulácia v Petriho sieti prebieha pomocou premiestňovania tokenov z miesta do miesta, ak to dovoľuje prechod. Bližšie informácie ohľadom Petriho sietí rozoberáme v kapitole (5).

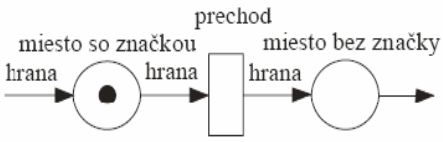
5 Petriho siete

Petriho siete (Petri Nets, ďalej PN) sú vhodným grafickým, ale aj matematickým nástrojom na modelovanie, analýzu a verifikáciu vlastností diskrétnych systémov, ktoré môžu byť [1]:

- Súbežné (concurrent)
- Asynchrónne
- Paralelné
- Distribuované
- Stochastické (nedeterministické)

PN predstavuje orientovaný bipartitný graf obsahujúci dva typy uzlov: miesta (places) a prechody (transitions), ktoré sú vzájomné poprepájané hranami (arcs).

Prívlastok orientovaný znamená to, že hrany v grafe sú orientované. Prívlastok bipartitný označuje stav, kde sa množina uzlov grafu skladá z dvoch vzájomne disjunktných podmnožín (množina miest a množina prechodov). Pri grafickej reprezentácii PN sa miesta znázorňujú ako krúžky a prechody ako obdĺžniky. Hrany spájajú miesto s prechodom alebo prechod s miestom. Značenie (marking), resp. stav priradí každému miestu nezáporné celé číslo. Ak značenie priradí miestu *p* nezáporné celé číslo *k*, hovoríme, že *p* je označené (marked) *k* značkami - tokenmi (tokens). Pri grafickej reprezentácii PN umiestnime do krúžku symbolizujúceho dané miesto *k* bodiek. Jednotlivé prvky PN sú zobrazené na obrázku č.1, pričom platí, že:



Obrázok č.1: Komponenty zovšeobecnenej Petriho siete.

- miesto môže obsahovať celočíselný nezáporný počet tokenov
- v okamihu aktivácie prechodu sa odoberú tokeny zo vstupných miest a pridajú sa na výstupné miesto prechodu
- orientované hrany prepájajú miesta a prechody
- počiatočné značenie (umiestnenie tokenov v miestach pred prvým preskokom) popisuje počiatočný stav systému

- vývoj systému je reprezentovaný presunom tokenov v sieti na základe aktivácie (prepálenia) prechodu
- každé nové značenie predstavuje nový stav [1].

5.1 Typy Petriho sietí

Existuje viacero rôznych rozšírení (modifikácií) Petriho sietí, ktoré boli vytvorené aby boli schopné jednoduchšie modelovať viacero oblastí praktického využitia. Petriho siete boli a stále sú upravované tak, aby vyhovovali potrebám systémov na ktoré nestačí opis iba pomocou základných Petriho sietí. Tieto zmeny a prídavné rozšírenia Petriho sietí nám slúžia na jednoduchšie a zrozumiteľnejšie modelovanie a opis zložitých a komplexných dynamických systémov.

Typy Petriho sietí rozlišujeme na dve základné skupiny a to PS nízkej úrovne a PS vyššej úrovne.

Petriho siete nízkej úrovne:

- C/E (Condition/Event) Petriho siete,
- P/T (Place/Transitions) Petriho siete,
- Deterministické časované (Timed) Petriho siete,
- Stochastické časované (Stochastic) Petriho siete.

Petriho siete vyššej úrovne:

- Farebné (Coloured) Petriho siete,
- Hierarchické (Hierarchical) Petriho siete,
- Fuzzy Petriho siete,
- Objektovo-orientované (Object-Oriented) Petriho siete.

Rôzne rozšírenia Petriho sietí môžu byť vzájomne kombinované aby sme dosiahli čo najlepšiu modelovaciu schopnosť. Petriho siete vyššej úrovne môžu byť napríklad rozšírené o časovú závislosť. Fuzzy Petriho siete sú založené na teórii fuzzy množín. Časovanú fuzzy Petriho sieť chápeme ako model neurčitých časových súvislostí. Tento typ siete tu nebude bližšie rozoberaný [5].

5.1.1 C/E Petriho siete

C/E sieť je najjednoduchší typ, ktorý sa dokonca v niektorých prípadoch chová ako konečný stavový automat. Je podtriedou P/T Petriho sietí. C/E - Condition/Event (podmienka/udalosť) znamená, že sú tieto siete modelované podmienkami a udalosťami, ktoré môžu nastať pri ich splnení.

Základné pravidlá:

- miesta siete reprezentujú určité podmienky,
- prechody siete reprezentujú udalosti, ktoré môžu nastať iba ak sú splnené všetky vstupné podmienky a zároveň nie sú splnené žiadne výstupné podmienky,
- po prevedení udalosti platí, že sú splnené všetky jej výstupné podmienky a nie je splnená žiadna vstupná podmienka,
- značky siete sa nachádzajú v miestach a reprezentujú platnosť podmienky (logický stav), pričom v každom mieste môže byť len jedna alebo žiadna značka,
- orientované hrany reprezentujú tok udalostí.

Celkový stav siete je daný množinou podmienok splnených v danom časovom okamihu. K zmenám stavov siete dochádza uskutočnovaním udalostí. Ak má udalosť splnené vstupné podmienky ešte to neznamená, že v tom okamihu musí nastať, tj. môže ale nemusí. Pri udalostiach rozlišujeme či sa vyskytujú nezávisle na sebe alebo v určitom poradí.

Špeciálne prípady C/E Petriho sietí:

1. *Konečný automat* - každý prechod siete má práve jednu vstupnú a jednu výstupnú podmienku a v sieti sa nachádza jediná značka, ktorá určuje aktuálny stav automatu,
2. *Čistá sieť* - sieť v ktorej žiaden prechod neobsahuje slučku (loop), čo je miesto, ktorého vstupná aj výstupná hrana patrí tomuto prechodu (spustením udalosti sa nemení platnosť podmienky),
3. *Elementárna sieť* - je to čistá sieť v ktorej má každý prechod aspoň jednu vstupnú a jednu výstupnú podmienku [5].

5.1.2 P/T Petriho siete

P/T - Place/Transition (miesto/prechod), znamená že sú definované miestami (stavy) a prechodmi (udalosti).

Pravidlá P/T sietí:

- miesta siete môžu mať udanú kapacitu ,ktorá úvádza maximálny možný počet značiek nachádzajúcich sa v danom mieste a je daná celým prirodzeným nezáporným číslom,
- ak miesto nemá udanú kapacitu má automaticky neobmedzenú kapacitu počtu značiek, hodnotu značenia miesta siete vyjadrujeme grafickým znázornením bodiek alebo pri väčšom počte celým nezáporným číslom,
- hrany siete môžu mať udanú váhu ,ktorá udáva násobnosť hrany a je rovnako daná celým nezáporným prirodzeným číslom,
- ak hrana nemá udanú váhu má automaticky váhu rovnú 1, prechod je spustiteľný ak každé jeho vstupné miesto obsahuje minimálne taký počet značiek aká je váha jeho hrany vstupujúca do daného prechodu a každé jeho výstupné miesto má taký počet značiek, že jeho zvýšenie o váhu jeho hrany vystupujúcu z prechodu neprevyšuje danú kapacitu miesta,
- po spustení prechodu sa z každého vstupného miesta presunie toľko značiek aké sú váhy ich výsupných hrán a naopak do výstupných miest prechodu sa pridajú značky zodpovedajúce váham hrán vstupujúcim do týchto miest.

P/T Petriho siete s prioritami:

- Každému prechodu siete môže byť priradená priorita celým nezáporným číslom. Spustenie prechodu má tak ďalšiu vstupnú podmienku. Súčastne môžu byť spustené len prechody s rovnakou prioritou inak je vždy spustený prechod s vyššou prioritou.

P/T Petriho siete s inhibítormi:

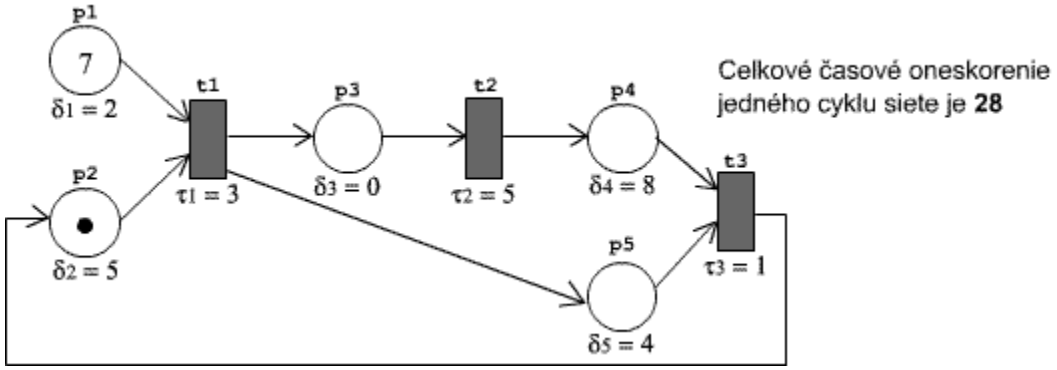
- Inhibítor je ďalší typ orientovanej hrany, ktorý je zakončený namiesto šípky kolieskom. Ich použitím podstatne zväčšujeme modelovaciu schopnosť Petriho siete až na úroveň Turingových strojov ale zároveň znižujeme možnosť ich analýzy. Je to vstupná testovacia hrana (podmienka) spustiteľnosti prechodu, ktorá je splnená ak miesto obsahuje menší počet značiek ako je váha inhibítora. Pri spustení prechodu sa z miesta, ktoré je s ním spojené inhibítorom neodoberajú žiadne značky narozdiel od štandardného vstupného miesta. Inhibítor je vždy možné použiť výhradne ako vstupnú hranu prechodu a nie ako výstupnú. P/T sieťe s inhibítormi sú z teoretického hľadiska schopné modelovať všetko čo je možné vyjadriť algoritmom [5].

5.1.3 Deterministické časované Petriho siete

Zapojením času do P/T Petriho sietí získame veľmi silný nástroj pre modelovanie dynamických systémov a ich analýzu výkonnosti. Čas je v prípade deterministických Petriho sietí vždy konštantný. Čas môže byť priradený všetkým prvkom Petriho siete a platí:

- priradenie času prechodom - po spustení prechodu zostávajú značky v prechode daný časový interval (pokiaľ daná udalosť neskončí),
- priradenie času miestam - značky zostávajú daný časový interval vo vstuptnom mieste prechodu od jeho spustiteľnosti po dobu trvania prechodu,
- priradenie času značkám - značky majú po prevední prechodu priradenú určitú časovú hodnotu počas ktorej nemôžu byť znova použité,
- priradenie času hranám - hranám je priradený daný časový interval prenosu značiek (vyjadruje rýchlosť prenosu) [3].

Na obrázku č.2 je sieť, v ktorej sú značkám a prechodom priradené časové oneskorenia. Časy pri miestach siete v tomto prípade reprezentujú časovú hodnotu počas ktorej nemôžu byť značky znova použité. Čas sa začína odratúvať hneď po vstúpení jednotlivých značiek do daného miesta. Sieť vykonáva opakovane 7-krát za sebou rovnaký cyklus udalostí pričom celkové oneskorenie jedného cyklu je 28.



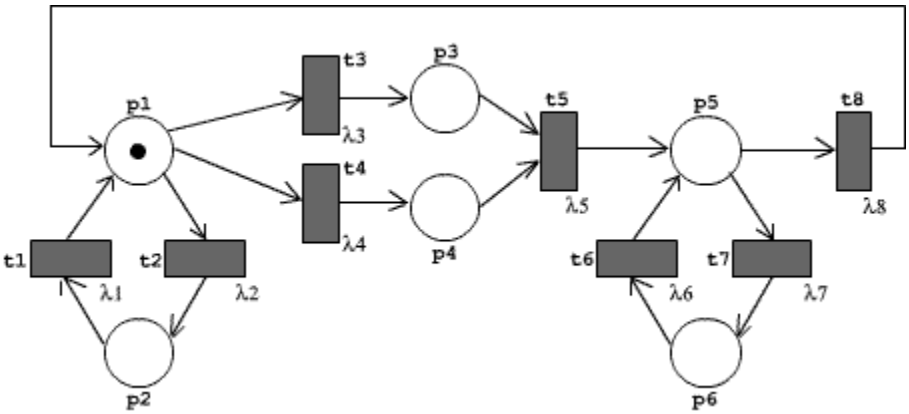
Obrázok č.2: Deterministická časovaná Petriho sieť.

5.1.4 Stochastické časované Petriho siete

Stochastický proces je taký proces v ktorom sú zmeny stavov podmienené vplyvom náhodných faktorov. V prípade stochastických Petriho sietí sú prechodom priradené početnosti a oneskorenia spúštania, ktoré reprezentujú určité náhodné veličiny. Časové oneskorenie môže byť priradené aj miestam siete. Početnosti spúštania prechodov majú náhodné exponenciálne rozloženie a vyjadrujú ako často je spustený prechod ak má stále splnené vstupné podmienky. Prevrátené hodnoty početností prechodov vyjadrujú čas vykonania udalostí, ktoré tieto

prechody reprezentujú. Počas vykonávania udalosti (prechodu), ktorý trvá náhodný čas čakajú značky vo vnútri prechodu. Stochastické PS sú rovnako využívané na analýzu výkonnosti systémov [5].

Na obrázku č.3 je stochastická Petriho sieť s priemernými početnosťami priradenými všetkým prechodom. Trvanie určitého prechodu je náhodná premenná s priemernou hodnotou rovnou 1 / λ_i . Každý prechod predstavuje činnosť, ktorá trvá náhodný čas, počas ktorého sú značky zo vstupných miest blokované v prechode.



Obrázok č.3: Stochastická časovaná Petriho sieť.

5.1.5 Farebné Petriho siete (CPN)

Ak chceme opísať reálny systém, ktorý pracuje so zložitými a viacerými dátovými typmi, potom nám už základné Petriho sieťe ako model nestačia lebo by boli príliš zložité a ťažkopádne a takýto opis je veľmi náročný. Pre opis zložitých manipulácií s dátami boli vyvinuté vysoko-úrovňové Petriho sieťe z ktorých sú najznámejšie a najpoužívanejšie farebné Petriho siete. Hlavnou myšlienkou je vzájomný pohyb viacerých typov farebne rozlíšených značiek v sieti.

Rozšírenia oproti P/T Petriho sieťam:

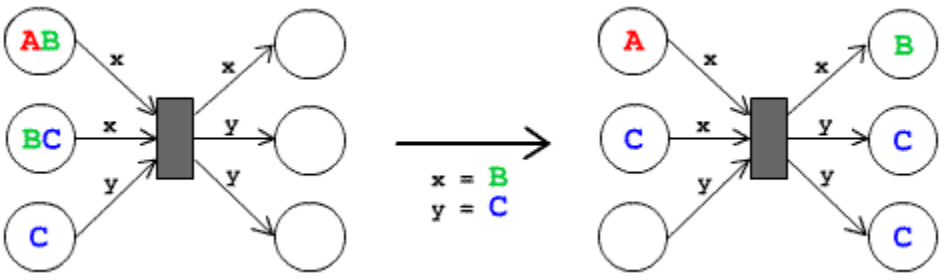
- sú rozšírené o viacero rôznych typov značiek siete, ktoré sú dané triedami farieb - sú farebne rozlíšené a značky rovnakej farby sú značky rovnakého typu,
- triedy farieb chápeme ako dátové typy, ktoré môžu byť konečné, nekonečné, jednoduché (elementárne) alebo zložité, (napr. výpočtové typy, rôzne druhy čísel, deň, rok, vektory,.. atď.), pričom každý dátový typ je daný množinou hodnôt,
- s dátovými typmi sú spojené najrôznejšie operácie, napr. číselné alebo logické operácie,
- každému miestu sú priradené typy značiek (triedy farieb), ktoré sa môžu v danom mieste nachádzať a stav miesta je daný multimnožinou jeho značiek, ktoré sa v ňom práve nachádzajú,

- každý prechod obsahuje navyše podmienku reprezentovanú výrazom vytvoreným z konštánt alebo premenných, ktorý po vyhodnotení dáva logickú hodnotu,
- z konštánt a premenných je taktiež vytvorený hranový výraz priradený hranám, ktorý po vyhodnotení dáva multimnožinu značiek typov prislúchajúcu miestu, ktoré patrí hrane.

Pre prechod farebnej PS platí:

- je spustiteľný ak je splnená jeho podmienka, ktorá po vyhodnotení dáva logickú hodnotu true,
- je spustiteľný ak multimnožina značiek každého vstupného miesta je rovnaká alebo väčšia ako multimnožina, ktorú dostaneme vyhodnotením hranového výrazu danej hrany, ktorá vedie z nášho vstupného miesta do prechodu,
- po jeho prevedení sa multimnožiny značiek vstupných miest zmenšia o multimnožinu, ktorú dostaneme vyhodnotením hranového výrazu vstupných hrán prechodu a naopak multimnožiny značiek výstupných miest sa zväčšia o multimnožinu, ktorá vznikne vyhodnotením hranového výrazu výstupnej hrany prechodu do daného miesta,
- pri oboch vyhodnoteniach hranového výrazu musia byť za premenné všade dosadené rovnaké hodnoty,
- prechod nemusí byť spustiteľný pri všetkých hodnotách dosadzovaných za premenné.

Jednoduchý príklad modelovania farebnej PS v ktorej sú hranám priradené premenné (x, y) a v miestach siete sú tri typy značiek alebo triedy farieb (A, B, C). K jednotlivým premenným priradíme typ značky podľa obrázka x=B, y=C. Ak by sme premenným priradili inú hodnotu napr. x=A, y=C prechod by nebol spustiteľný lebo by neboli splnené vstupné podmienky. Po spustení prechodu sa značky presunú do výstupných miest a výsledné značkovanie vidíme na obrázku [5].



Obrázok č.4: Jednoduchá farebná Petriho sieť.

5.2 Vlastnosti Petriho sieti

Reprezentujú dôležité problémy a črty základných P/T (Place-Transition) sietí riešené u všetkých formálnych modelov. Pomocou P/T Petriho sietí môžeme modelovať pomerne širokú triedu problémov, a zároveň môžeme takéto modely jednoducho analyzovať a určovať ich vlastnosti. Aby sme vedeli model systému vhodne analyzovať, musia byť jeho vlastnosti rozhodnuteľné. Teoretické podklady sú čerpané z [5]

5.2.1 Vlastnosti správania sa PS

Opisujú Petriho sieť z hľadiska jej správania a závisia od počiatočného značkovania siete M_0.

- dosiahnuteľnosť značkovania
- ohraničenosť
- bezpečnosť
- živosť
- pokrytie
- bezkonfliktnosť
- bezkontaktnosť

Mnohé dôležité vlastnosti Petriho sietí možno analyzovať s využitím grafov dosiahnuteľnosti a pokrytia. Graf dosiahnuteľnosti Petriho siete je spätý s dosiahnuteľnosťou.

5.2.2 Vlastnosti štruktúry PS

Opisujú Petriho sieť z hľadiska jej štruktúry a nezávisia od počiatočného značkovania siete M_0. Štruktúrou je myslený návrh (konštrukcia) siete - poprepájanie základných stavebných prvkov PS a počiatočné označkovanie.

- reverzibilnosť (konzistentnosť)
- konzervatívnosť

Ďalšie možnosti štruktúrnej analýzy Petriho sietí predstavujú metódy, ktoré sú založené na lineárnej algebraickej reprezentácii, použití matíc a riešení sústav rovníc. Sú to P-invarianty a T-invarianty a ide o štruktúrne vlastnosti. P-invariant je množina miest, ktoré nemenia svoje značky v priebehu spúštania prechodov. Ich identifikovanie nám pomáha v analýze živosti PS a v zisťovaní určitých špecifických vlastností, ktoré má modelovaný systém zachovávať. T-invariant udáva pri určitom značkovaní PS koľkokrát je potrebné spustiť každý prechod siete aby sme

znovu dosiahli toto značenie. Maticová reprezentácia zjednodušuje prácu s pojmami, ktoré súvisia zo značkovaním PS, a ktoré sa úzko dotýkajú chovania modelovaného systému. Tieto metódy si tu len veľmi stručne opíšeme.

5.2.3 Dosiahnuteľnosť značkovania a pokrytie

Počiatočné značkovanie siete M_0 je dané definíciou Petriho siete. Každý diskrétny udalostný systém začína svoju činnosť určitým počiatočným stavom. Otázka či je značenie dosiahnuteľné z počiatočného značenia, teda či je prvkom stavového priestoru PS, patrí medzi základné problémy analýzy Petriho sietí.

Značkovanie M_i je dosiahnuteľné zo značkovania M_j , keď M_i získame M_j realizáciou konečného počtu *k* priechodov. Znamená to, že existuje určitá postupnosť spustenia prechodov pri ktorej sa dostaneme z jedného značkovania siete do iného. V prípade že, $k = 1$, hovoríme o bezprostrednej dosiahnuteľnosti. Množinu všetkých značkovaní dosiahnuteľných z M_0 označujeme ako $R(M_0)$.

Graf dosiahnuteľnosti

Množinu dosiahnuteľných značkovaní a postupnosti spúšťania prechodov pre danú Petriho sieť vieme vizualizovať graficky pomocou orientovaného grafu. Slúži ako prehľadná pomôcka pri analýze viacerých vlastností Petriho siete. Graf má nasledujúce vlastnosti:

- množina $R(M_0)$ je množina všetkých vrcholov grafu,
- hrany reprezentujú spustenie konkrétneho prechodu,
- pri dosiahnutí duplicitného značkovania vedie hrana grafu k prvému výskytu tohto značkovania (systém sa tzv. vráti do určitého značkovania v ktorom sa už nachádzal),
- M_0 je počiatočný vrchol grafu.

Strom dosiahnuteľnosti

Postup vytvárania je podobný, s tým rozdielom, že pri dosiahnutí duplicitného značkovania sa tento vrchol označí ako koncový inou farbou a nepokračuje z neho žiadna hrana. Na začiatku máme množinu odsahujúcu jeden prvok a to značkovanie M_0. Následne do množiny pridáme všetky bezprostredne dosiahnuteľné značkovania k značkovaniam, ktoré už máme v množine. Tretí krok je zistenie, či sa nám množina pridaním určitého značkovania zväčšila. Ak áno tak postupujeme znovu druhým krokom a ak nie znamená to, že sa značkovanie opakuje a nepokračujeme ďaľej.

Pokrytie

Značkovanie M_i je pokryté značkovaním M_j, ak je od neho menšie alebo mu je rovné, tj. platí $M_i \leq M_j$.

Určenie množiny dosiahnuteľných značkovaní pre neohraničenú (nekonečnú) Petriho sieť konečným spôsobom je tiež možné. Množina dosiahnuteľných značkovaní môže byť nekonečná, čo nastáva ak niektoré miesta v sieti môžu neobmedzene zväčšovať svoj počet zančiek (tokenov). Na vyjadrenie nekonečnosti sa používa symbol ω.

Graf pokrytia (strom pokrytia)

Pre neohraničené PS môžeme skonštruovať graf alebo strom pokrytia namiesto grafu dosiahnuteľnosti. Umožňujú reprezentovať množinu všetkých dosiahnuteľných značkovaní pomocou konečného počtu rozšírených značkovaní, ktoré pokrývajú všetky dosiahnuteľné značkovania v danej PS.

5.2.4 Ohraničenosť a bezpečnosť

Ohraničenosť miesta a siete

Pri opise správania sa systémov zisťujeme, či má špecifikovaný systém konečnú množinu všetkých stavov. V Petriho sieťach hovoríme o tejto vlastnosti ako o ohraničenosti siete. Miesto p je ohraničené vtedy, ak jeho počet značiek neprekročí určitú konečnú hodnotu *k* (k-ohraničené miesto). PS je k-ohraničená ak pri každom dosiahnuteľnom značkovaní sú všetky jej miesta maximálne k-ohraničené. To tiež znamená, že PS je ohraničená len ak je jej množina dosiahnuteľných značkovaní konečná. Ohraničenosť sieťe je požadovaná pri väčšine modelovaných systémov.

Bezpečnosť miesta a siete

PS sa nazýva bezpečná, ak je 1-ohraničená. To znamená, že počet značiek vo všetkých miestach siete musí byť 0 alebo 1 pri všetkých dosiahnuteľných značkovaniach. Bezpečnosť sieťe je požadovaná zväčša pri modelovaní logických podmienok v systémoch.

5.2.5 Živosť a reverzibilnosť

Jedným zo základných problémov pri modelovaní udalostných systémov je vyvarovať sa takému stavu systému z ktorého nie je možné spustiť žiaden prechod PS čo znamená že je systém nečinný a nemôže v ňom nastať žiadna udalosť. Tento stav nazývame mŕtvy stav alebo uzamknutie (deadlock). Nastáva napríklad v situácii pri zdieľaní zdrojov, kde proces čaká na pridelenie zdroja, ktorý už bol pridelený inému procesu a podobne tento proces čaká na pridelenie zdroja, ktorý už bol pridelený prvému procesu, čo znamená, že sa navzájom blokujú a systém sa uzamkne.

Živosť

PS je živá ak je každý jej prechod živý a pri činnosti systému, tj. pri spúšťaní prechodov siete, je vždy možné každý prechod spustiť znovu - nestráca možnosť, že by už nebol v budúcnosti spustený. Sieť je živá ak neobsahuje žiaden mŕtvy stav. Živosť siete závisí od počiatočného značkovania M_0. Dosiahnuteľné značkovanie je mŕtve ak z neho nie je možné spustiť žiaden prechod. Ak sú všetky dosiahnuteľné značenia z M_0 živé, tak aj PS je živá.

Prechod v Petriho sieti je živý vtedy, ak pre každé jeho značkovanie M_i, dosiahnuteľné z M_0 existuje také značkovanie M_j dosiahnuteľné z M_i, z ktorého je priechod t spustiteľný.

Rozlišujeme viacero úrovní živosti prechodu PS:

- **L0-živý** (mŕtvy) : ak nie je spustiteľný v žiadnom z dosiahnuteľných značkovaní,
- **L1-živý** : pokiaľ môže byť spustený aspoň raz - aspoň v jednom zo značkovaní,
- **L2-živý** : ak pre určité celé kladné číslo *n* môže byť v nejakej postupnosti značkovaní spustení aspoň *n*-krát,
- **L3-živý** : ak je spustiteľný nekonečný počet krát v určitej postupnosti značkovaní,
- **L4-živý** (živý) : ak pre každé zo značkovaní existuje postupnosť spustenia prechodov, pričom daný prechod je v postupnosti spustený aspoň raz.

PS je L_i-živá, ak sú všetky jej prechody L_i-živé.

Reverzibilnosť

Táto vlastnosť vyjadruje cyklický charakter správania sa systému. Veľa systémov totiž pracuje opakovaným vykonávaním určitej postupnosti udalostí (procesov). To znamená, že po ukončení vykonávanej postupnosti vždy vráti systém do počiatočného stavu pre opakovanie.

PS je reverzibilná ak je vždy dosiahnuteľné počiatočné značkovanie. Platí, že z každého dosiahnuteľného značkovania siete je dosiahnuteľné počiatočné značkovanie M_0.

Cyklická vlastnosť správania sa je využívaná najmä vo výrobných systémoch.

5.2.6 Bezkonfliktnosť

Konflikt v PS nastáva, ak spustenie jedného prechodu siete znižuje stupeň spustiteľnosti iného prechodu siete alebo jeho spustenie tohto prechodu úplne znemožnuje. PS nazývame bezkonfliktnou ak v nej nemôže nastať žiadny takýto konflikt, tj. nemôže sa nachádzať v stave v ktorom sú viaceré prechody spustiteľné ale spustiteľnosť jedného zníži spustiteľnosť iného.

Rozlišujeme dva typy konfliktu:

- **symetrický** - ak prevedením jedného prechodu znížim spustiteľnosť druhého a aj naopak prevedením druhého ovplyvním prvý,
- **asymetrický** - ak prevedenie jedného prechodu znižuje spustiteľnosť druhého ale druhý neovplyvňuje spustiteľnosť prvého.

5.2.7 Konzervatívnosť

Časté použitie PS je aj pri modelovaní pridelenia rôznych systémových zdrojov v systéme. PS je striktne konzervatívna, ak celkový počet zdrojov (značiek) v počiatočnom značkovaní je rovnaký aj vo všetkých ostatných dosiahnuteľných značkovaniach. Počet značiek v PS ostáva po celú dobu činnosti systému nemenný.

PS je konzervatívna vzhľadom na váhový vektor *v*, ak platí, že súčin $M_0.v = M_i.v = ... = M_j.v$, kde (M_i, ... , M_j) je množina všetkých dosiahnuteľných značkovaní danej siete. O takejto PS môžeme hovoriť napríklad ak z jedného stavu (procesu) siete vzniknú po spustení prechodu dva a viac stavov (procesov), ktoré sa znovu spoja do jedného spoločného procesu.

Táto vlastnosť býva modelovaná najmä v distribuovaných počítačových systémoch, kde vzniká problém dostupnosti procesora a iných častí počítača a problém vzniku kritických sekcií.

6 Problematika lokalizácie porúch v komunikačných systémoch

Diagnostika, detekcia porúch a ich separácia sú veľmi dôležitými a náročnými úlohami v oblasti automatického riadenia veľkých komplexných systémov. Tieto úlohy v minulosti boli a v súčasnosti stále sú výzvou pre mnohých výskumných pracovníkov v oblasti teórie systémov a ich riadenia, prípadne tiež návrhu.

Komplexnosť komunikačných sietí spôsobuje, že prevencia porúch v zmysle zamedzenia ich vzniku a výskytu nie je možná (porúch hardvérového i softvérového charakteru, ale rovnako ako aj porúch spôsobených ľudskými chybami) – o to je dôležitejšia práve rýchla detekcia vzniknutých porúch, ich presná lokalizácia a následná izolácia/odstránenie. Rozsiahlosť existujúcich komunikačných sietí navyše vnáša potrebu maximálnej automatizácie tohto procesu.

Lokalizácia porúch funguje na základe zberania, klasifikácie a analýzy chýb. Vhodne fungujúci detekčný systém dokáže z rôznej konštelácie zaznamenaných chýb identifikovať a lokalizovať poruchu, ktorá je inak v mieste výskytu nebadateľná, respektíve nie je možné ju priamo lokalizovať na základe napríklad samostatných stavových informácií. [6]

Na riešenie takýchto úloh boli navrhnuté viaceré rôzne modely, prístupy a formalizmy, napr. stavovo priestorové metódy (state-space), metódy detekcie a izolácie porúch, poruchové stromy (fault trees), invariantov, použitie prostriedkov umelej inteligencie, a i.

6.1 Detekcia a izolácia porúch

Najpoužívanejšie techniky sú z tejto kategórie – sú najzrozumiteľnejšie a najjednoduchšie na pochopenie. Ideou je imitácia postupu skutočného človeka, ktorého schopnosti vychádzajú na jednej strane z hlbokých vedomostí o komunikačných sieťach a na druhej z praktických skúseností nadobudnutých praxou. Lokalizačné systémy najčastejšie používajú informačnú základňu reprezentovanú množinou pravidiel, ktoré sa v cykloch zväzujú do postupností podľa aktuálnych posudzovaných podmienok. V systémoch s malým rozsahom môže byť táto technika účinná – skúsenejší profesionál dokáže vymyslieť množinu pravidiel, ktorá pokryje najčastejšie poruchy (čiže ide viac o „praktický“ prístup, ako „vedecký“).

6.2 Techniky prechádzania modelu

Tieto techniky využívajú formálnu reprezentáciu komunikačného systému s jasne definovanými vzťahmi a spojeniami medzi komponentami siete (entitami tvoriacimi danú komunikačnú sieť). Proces lokalizácie poruchy začína v mieste hlásenej chyby (zistenej nezrovnalosti zariadením v komunikačnej sieti v monitorovaných veličinách a stavoch), nasleduje identifikovanie všetkých ďalších súvisiacich chýb a pomocou takto vytvoreného súboru sa identifikujú poruchové elementy v sieti.

6.3 Grafovo teoretické techniky

Tieto techniky sú postavené na grafovom modeli systému, konkrétne na smerovom modeli propagácie porúch, ktorý popisuje, ako a kde sa konkrétna porucha prejaví. Zostavenie takéhoto modelu vyžaduje presné a podrobné informácie o závislostiach medzi jednotlivými komponentami siete a od jeho presnosti sú následne priamo úmerne závislé aj výsledky lokalizácie porúch. V praxi sa rozsiahlosť redukovať potrebou modelov propagácie porúch len pre práve existujúce spojenia. Zaujímavá je možnosť viacerých typov porúch jedného objektu v prípade použitia kauzálnych grafov namiesto závislostných (napríklad kompletné znefunkčnenie, dlhá odozva, vysoká stratovosť paketov...) - takto popisované stavy sa približujú reálnemu systému fungujúcemu v reálnom svete oveľa viac, ako predchádzajúce techniky, žiaľ opäť za cenu citeľne zložitejšieho návrhu a zostavenia modelu.

7 Modelovanie Petriho sietí

Cieľom nášho tímového projektu je vytvoriť model sieťového protokolu. Vytváranie komplexného modelu Petriho siete bez pomoci softvérových aplikácií v dnešnej dobe je ťažko realizovateľné. Potreba vytvoriť presný, stabilný a dostatočne funkčný model nás priviedla k výberu vhodného programového nástroja.

Hlavnými kritériami sú pre naše účely práce je dostatočná základná funkčnosť, stabilná verzia programu, dostatočná manuálová podpora, prehľadnosť používateľského prostredia, cenová dostupnosť, prípadne kompatibilita s operačným systémom Windows.

Konkrétne požiadavky sa budú ešte časom upravovať, no už teraz je zrejmé, že budeme potrebovať nástroj, ktorý bude obsahovať grafické používateľské rozhranie. Na pohodlné modelovanie, rýchly návrh a jednoduché úpravy návrhu by textový mód prinášal zbytočné komplikácie. Drvivá väčšina dnes dostupných nástrojov už grafický editor v sebe zahŕňa, takže s touto požiadavkou by nemal byť problém.

Podpora základných vlastností Petriho sietí ako je používanie miest a prechodov, ich animácia pri spustení simulácie musí byť zabezpečená. Jedná sa o základné vlastnosti, z ktorých náš model bude vychádzať.

Farebné Petriho siete v dnešnej dobe nie sú programovo dostatočne dobre zastúpené, takže táto funkcionalita nie je pre nás kľúčová. Samozrejme niektoré nové verzie programov túto možnosť ponúkajú, ale na úkor stability softvéru. To je v rozpore s našimi hlavnými kritériami, čiže túto možnosť uvítame len v prípade stabilnej verzie programu.

Časové Petriho siete poskytujú rozsiahlejšie možnosti modelovania a špecifickejšie nastavenia pri práci s časom. V našom prípade sa jedná o veľmi vhodnú funkciu, avšak taktiež nepatrí medzi hlavné kritériá. V budúcom riešení projektu sa možno prikloníme k programom s podporou časových Petriho sietí, avšak v tomto momente to prijímame len ako potenciálne využiteľnú funkčnosť navyše.

Hľadanie chýb v návrhu siete, resp. v jej simulácii poskytujú väčšinou len podnikové riešenia, preto na túto možnosť nespoliehame a simulovanie chýb bude vytvárané v našej réžii.

Poslednou veľmi významnou črtou programu musí byť cena. Možné alternatívy pre našu prácu sú licencie freeware, OpenSource alebo akademická verzia programu prípadne demo.

7.1 Programové nástroje

Výber konkrétnych nástrojov spočíval v prehľadávaní internetu a špecializovaných stránok, ktoré sa venujú modelovaniu Petriho sietí [7]. Taktiež podstatné boli odporúčania našej vedúcej projektu Ing. Jany Flochovej, Phd. , ktorá má v danej problematike dlhoročné skúsenosti a pracovala s rôznymi nástrojmi tohto typu.

Po konzultáciách a porovnaniach jednotlivých nástrojov sme do užšieho výberu vybrali štyri programy, ktoré spĺňajú nami vyššie spomínané požiadavky. CPN Tools udržiavaný univerzitou v Dánsku, Platform Independent Petri Net Editor 2 vytvorený na univerzite v Londýne, Netlab pochádzajúci z univerzity v Nemecku a HPSim, ktorý je vyvíjaný súkromne. Takže väčšinou sa jedná o akademické produkty, ktoré prešli dlhodobým vývojom.

7.1.1 CPN Tools (Coloured Petri Nets)

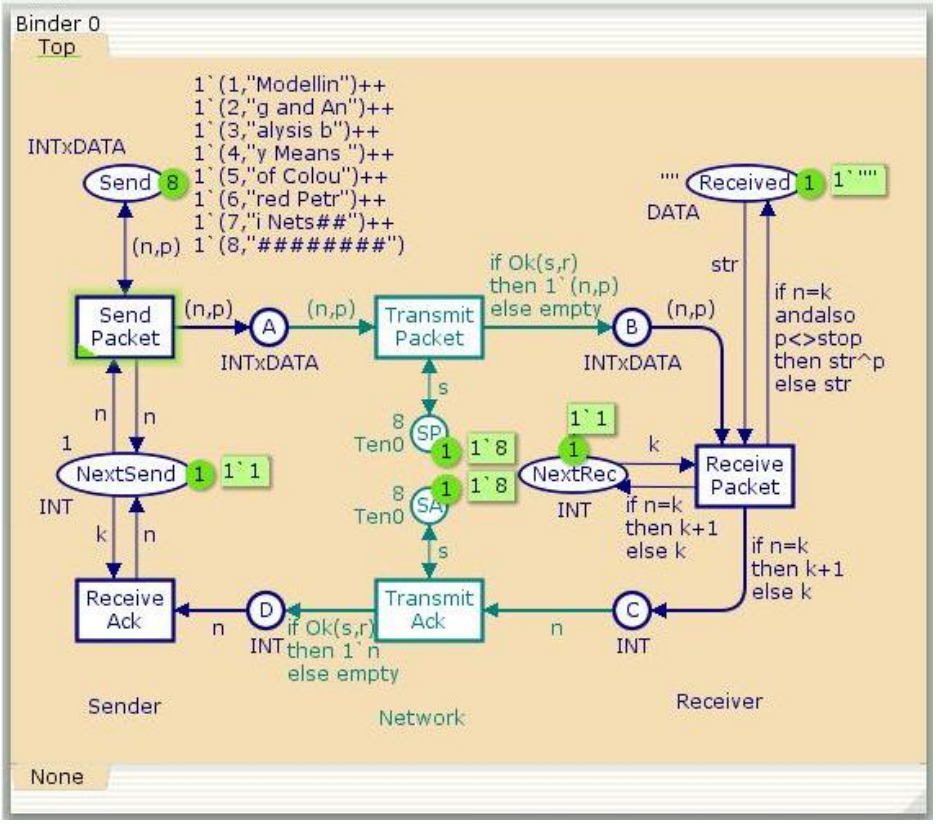
http://wiki.daimi.au.dk/cpntools/cpntools.wiki

Už z názvu vyplýva, že tento nástroj sa bude prevažne zameriavať na modelovanie, simuláciu a analýzu farebných Petriho sietí. Nahradil dlhé roky vyvíjaný CPN/Design Tool nástroj, s ktorým je kompatibilný.

Poskytuje prehľadné grafické editačné prostredie. Obsahuje analytické mechanizmy na zistenie základných parametrov siete ako je živosť, ohraničenosť, výkonnosť a pod. Samotná analýza sa rozdeľuje na analýzu simulácie a čiastkovú alebo úplnú analýzu miest a stavov. Taktiež kontrolné mechanizmy pri vytváraní siete ako je kontrola generovanej syntaxe a závislosti medzi elementmi.

Obsahuje podporu pre časované, nečasované ako aj hierarchické Petriho siete. Neobsahuje však podporu inhibítorov.

Simulátor má veľmi dobre spracované manuálové stránky s komplexným popisom funkcií programu spolu s názornými obrázkami. Dostupné sú aj rôzne príklady modelov vytvorených Petriho sietí.



Obrázok č.1: Model jednoduchého protokolu v nástroji CPN Tools.

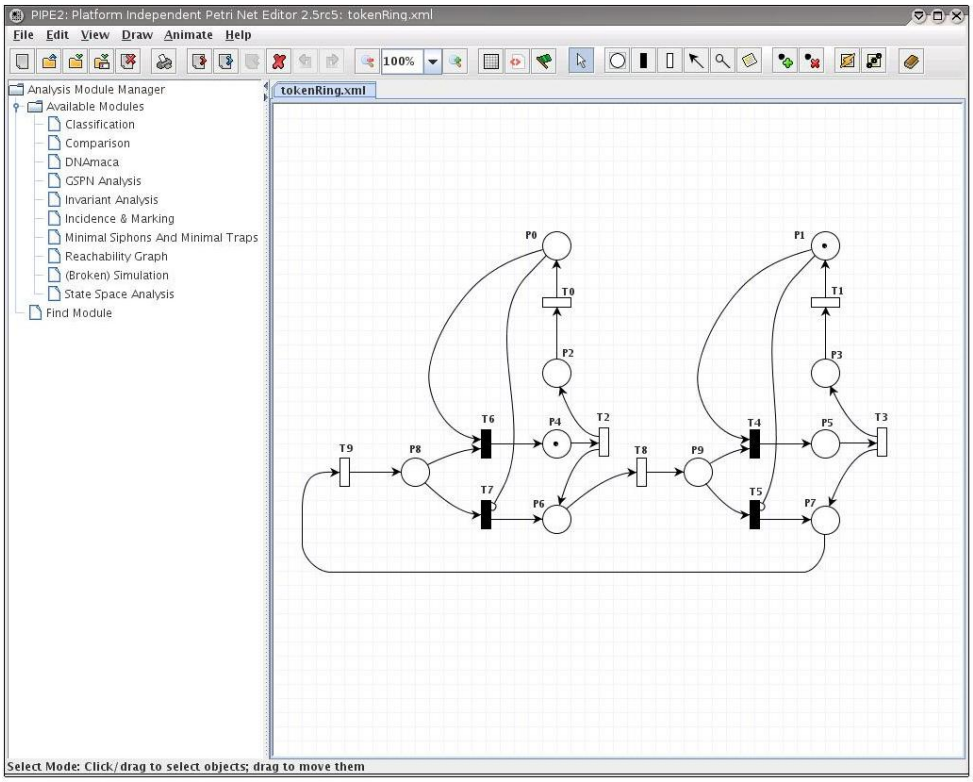
7.1.2 Platform Independent Petri Net Editor 2 (PIPE2)

http://pipe2.sourceforge.net/

Rýchly, výkonný a efektívny nástroj na modelovanie Petriho sietí vytvorený v prostredí multiplatformového jazyka Java. Poskytuje viaceré úrovne analýzy - analýza invariantov, analýza miest a stavov a analýza simulácie. Umožňuje tiež dotvoriť vlastné analytické moduly.

Jednoducho ovládateľné grafické rozhranie poskytuje rozsiahle možnosti nastavení, bezproblémové ovládanie a úpravu modelov. Animácia simulácie prehľadne znázorňuje časové správanie sa modelu. Obsahuje podporu pre stochastické Petriho siete. Generuje grafy dosiahnuteľnosti a incidenčné matice.

Na základe práce [8], je tento program odporúčaný aj našou vedúcou tímového projektu.



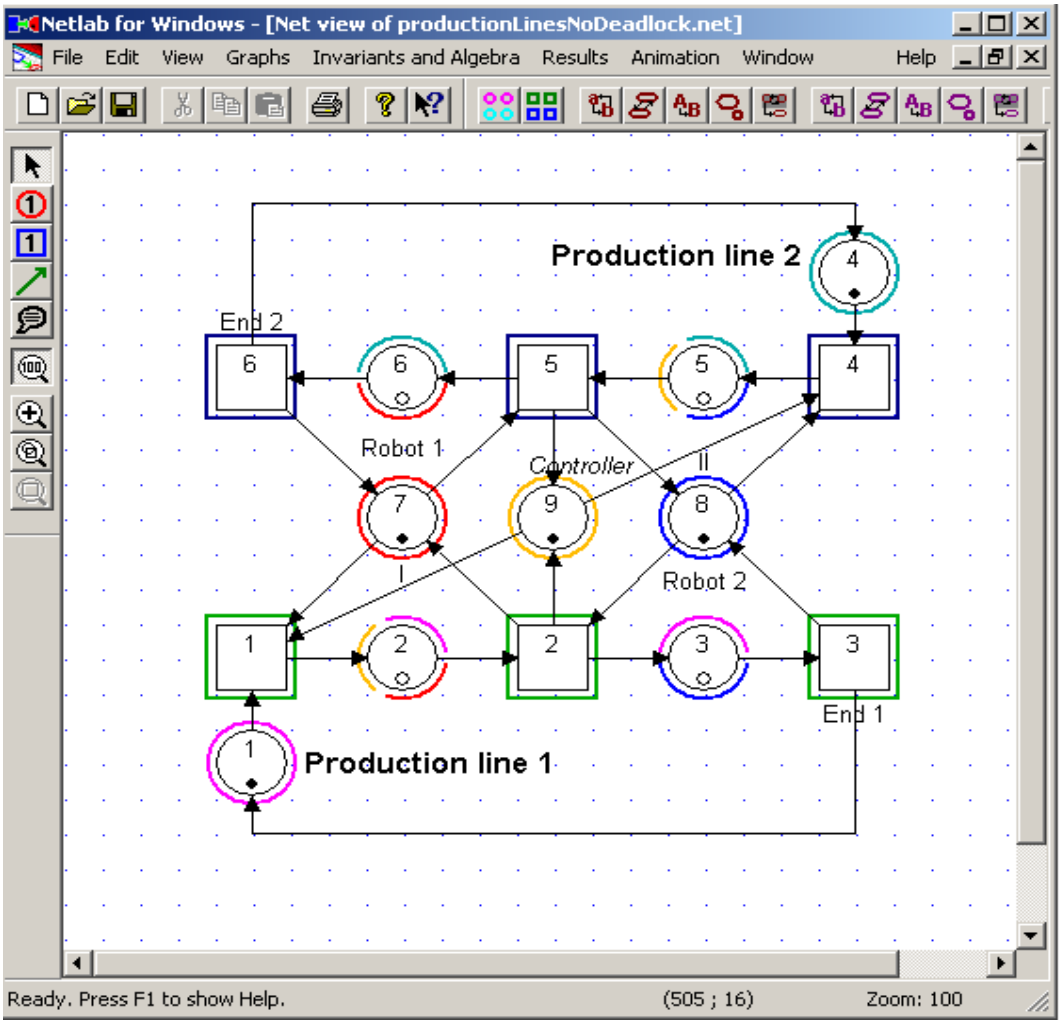
Obrázok č.2: Model token ring v programe PIPE2.

7.1.3 Netlab

http://www.irt.rwth-aachen.de/en/downloads/petri-net-tool-netlab.html

Simulačný nástroj na implementáciu Petriho sietí z dielne univerzity v Aachene. V manuálových stránkach uvádza možnosť diskrétneho modelovania stavového modelu Petriho siete nazývaného ako signalizačné Petriho siete. Podporuje prácu s invariantami miesta aj prechodu. Taktiež štruktúrovaná analýza umožňuje získavanie potrebných informácií o správaní sa siete, inhibítoroch a grafoch. Grafické prostredie a intuitívny editor vytvárajú jednoduchú cestu k návrhu simulačného modelu. Paralelné spracovanie viacerých modelov urýchľuje prácu a umožňuje sledovať vzájomné prepojenia sietí, resp. porovnávať výkonnosti jednotlivých modelov.

Za značnú nevýhodu považujeme to, že sa nám nepodarilo nájsť podrobné manuálové stránky a popis všetkých funkcií podporovaných týmto programom.



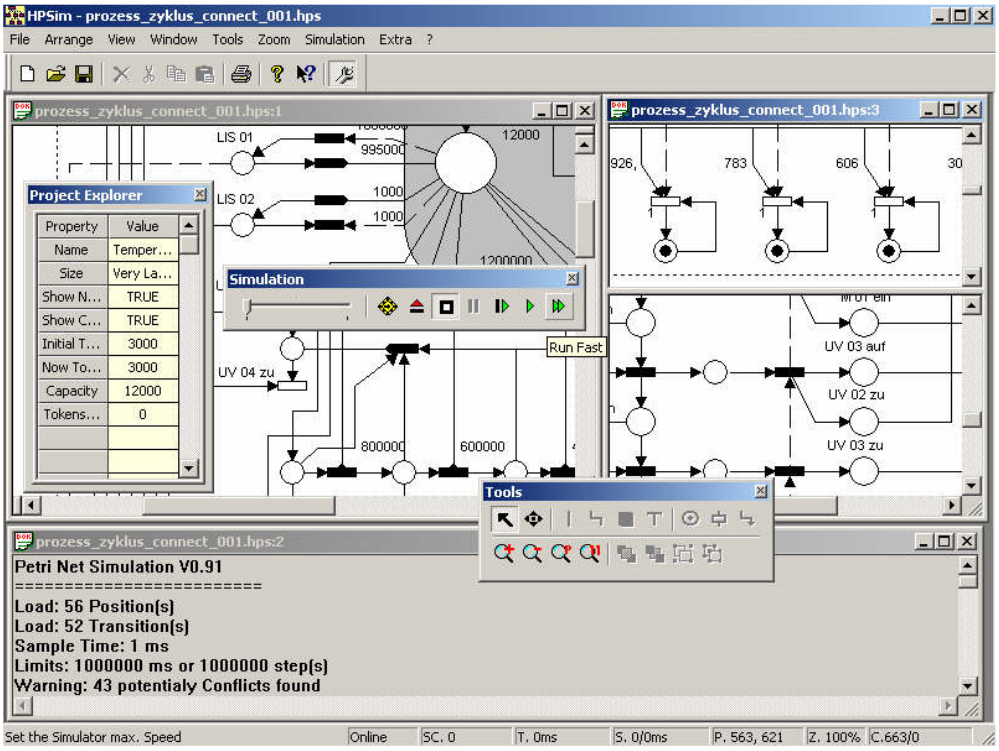
Obrázok č.3: Model produkčnej linky v programe Netlab.

7.1.4 HPSim

http://www.winpesim.de/3.html

Grafický editor Petriho sietí určený na vyučovacie účely, resp. pre užívateľov, ktorí sa chcú oboznámiť s funkcionalitou Petriho sietí. Umožňuje modelovanie stochastických aj časových sietí, váhové ohodnotenie hrán, časové prechody a pod. Grafický editor ponúka všetky štandardné funkcie na vytváranie a editáciu Petriho siete. Jednoduché rozhranie prináša komfort pre začiatočníkov aj pokročilých užívateľov.

Nevýhoda je nedostupnosť anglickej používateľskej príručky.



Obrázok č.4: Model systému v prostredí HPSim.

7.1.5 Zhodnotenie modelovacích nástrojov

Z vyššie analyzovaných nástrojov sme vybrali dva najvhodnejšie pre našu prácu. Prihliadali sme hlavne na funkcionalitu, efektívnosť ovládania a dostupnosť manuálových stránok o programe.

Do užšieho výberu sa dostali Platform Independent Petri Net Editor 2 a CPN Tools. Obidva poskytujú dostatočnú funkcionalitu a sú kvalitne zdokumentované. Avšak líšia sa v podpore pre určité vlastnosti Petriho sietí.

PIPE 2 umožňuje vytvárať modely stochastických sietí, aplikovať invarianty a rozsiahle možnosti analyzovania siete.

CPN Tools zase dokáže pracovať s farebnými aj časovými Petriho sieťami. Patrí medzi najpoužívanejšie prostriedky na modelovanie, poskytuje rozsiahlou podporou.

Ostáva teda zvážiť, ktorý z dvoch vybratých nástrojov bude vhodný pre náš typ Petriho siete. Ak budem potrebovať farebnú prípadne časovú Petriho sieť, odporúčaný nástroj je CPN Tools, ak nám postačí bežný typ siete s inhibítormi, tak program PIPE2 je vhodnejšia voľba.

Nasleduje prehľadová tabuľka s porovnaním jednotlivých nástrojov podľa [9].

	CPN Tools	**Platform Independent Petri Net Editor 2**	**Netlab**	**HPSim**
Licencia	Voľne dostupné	Voľne dostupné	Akademická	Voľne dostupné
Podpora typov Petriho sietí	Štandardné Časové	Štandardné	Štandardné Diskrétne	Štandardné Stochastiké Časové
Vlastnosti nástroja	Grafický editor Animácie Rýchla simulácia Stavy prechodov Analyzovanie Univerzálny výstup	Grafický editor Animácie Rýchla simulácia Stavy prechodov Zhustené stavy prechodov Univerzálny výstup Invarianty miest Invarianty prechodov Štrukturálna analýza Analytické moduly	Grafický editor Animácie Rýchla simulácia Stavy prechodov Analyzovanie Zhustené stavy prechodov Univerzálny výstup Invarianty miest Invarianty prechodov	Grafický editor Animácie Rýchla simulácia Analyzovanie Univerzálny výstup
Platforma	PC,Linux PC, MS Windows	Java	PC, MS Windows	PC, MS Windows

Tabuľka č.1: Porovnanie simulačných nástrojov.

8 Protokol OSPF

Open Shortest Path First (OSPF) je protokol, ktorý bol vytvorený organizáciou IETF v rokoch 1988 až 1991. Vznikol za účelom zavedenia vysoko spoľahlivého, funkčného a od výrobcov nezávislého štandardu na smerovanie paketov IP protokolu. V súčasnosti je OSPF používaný ako smerovací protokol vnútri autonómnych systémov (AS), preto ho radíme medzi tzv. Interior Gateway protokoly (IGP). Prvá verzia OSPF bola popísaná v RFC 1131. Neskôr bola nahradená OSPF verziou 2 publikovanou v RFC 1247 a jej rozšíreniami (RFC 1583, 2178 a 2328). Najnovšia verzia OSPF v3 podporujúca štandard IPv6 je definovaná v RFC 2740 [10].

OSPF patrí medzi Link State smerovacie protokoly. Tie vytvárajú v pameti smerovača kompletnú mapu siete označovanú ako topologická databáza (taktiež známa pod názvom Link State Database – LSDB). Nad touto databázou sa vykonávajú výpočty pomocou algoritmu zvaného Shortest Path First (SPF), ktorý hľadá najvýhodnejšie cesty do oznamovaných sieťových destinácií. SPF používa Dijkstrov algoritmus, ktorý je z hľadiska procesorového výkonu výpočtovo náročný. Preto OSPF protokol obsahuje mechanizmy na zamedzenie príliš častého spúšťania prepočtu (napr. Hold time – minimálny čas medzi dvomi po sebe idúcimi kalkuláciami SPF).

8.1 Základná funcionalita OSPF smerovačov

V tejto časti si popíšeme správanie smerovačov pracujúcich s protokolom OSPF [11].

- Smerovač vysiela v pravidelných intervaloch tzv. Hello pakety do priamo pripojených sietí a snaží sa nadviazať susedstvo s okolitými OSPF smerovačmi. V prípade, že sa smerovače zhodnú v požadovaných parametroch, stanú sa susedmi a naďalej si vymieňajú Hello pakety, čím udržujú svoje susedstvo aktívne.
- Aby mohlo dôjsť k výmene smerovacích informácií pomocou paketov Link State Advertisements (LSA), susedné smerovače musia nadviazať priateľstvo (tzv. adjacency). Smerovače dosiahnú finálny status priateľstva označovaný Full až keď majú zosynchronizované databázy LSDB.
- Smerovače si ukládajú prijaté LSA správy do LSDB a zároveň ich preposielajú svojim susedom. Týmto je zabezpečený konzistentný pohľad na topológiu siete pre každý OSPF smerovač.
- Keď topologická tabuľka (LSDB) smerovača obsahuje všetky potrebné informácie, spustí prepočet Dijkstrovho algoritmu SPF.
- Výsledkom prepočtu sú najkratšie a bezslučkové (loop-free) cesty do sieťových destinácií.

Ak v sieti nastane zmena (napr. linka sa stane nedostupnou), dotknutý smerovač pošle svojim susedom LSA správu o tejto zmene. Každý smerovač v sieti, ktorý obdržal túto informáciu musí spustiť SFP algoritmus a prepočítať novú cestu do danej siete, alebo sieť vyradiť zo smerovacej tabuľky, ak neexistuje náhradná cesta.

8.2 Typy sietí v OSPF

Protokol OSPF rozpoznáva niekoľko typov sietí:

- **Broadcast**

Ide o siete, ktoré dokážu prepojiť viac ako dva uzly v sieti a výslané rámce môžu prijímať súčasne všetky z týchto uzlov. Táto vlastnosť značne ovplyvňuje spôsob vytváranie priateľstiev (adjacencies) medzi smerovačmi o čom si povieme neskôr. Príkladom broadcastových sietí sú Ethernet a FDDI.

- **Point-to-point**

Siete, ktoré prepájajú dva smerovače sériovou linkou. Smerovače sa stávajú vždy priateľmi (adjacent).

- **Non Broadcast Multi Access (NBMA)**

Sieť tohto typu dokáže prepojiť viac smerovačov, avšak nedokáže posielať broadcasty. Nie je teda možné, aby všetky smerovače prijali vyslanú správu. Príkladom takejto siete je Frame Relay.

- **Point-to-multipoint**

Ide o špeciálny prípad NBMA siete, v ktorej sú všetky linky chápané ako point-to-point liky.

8.3 Typy OSPF paketov

Protokol OSPF používa 5 rôznych typov správ na komunikáciu medzi smerovačmi. Tieto správy sú enkapsulované priamo do IP protokolu, pričom v hlavičke paketu je v poli IP Protocol vždy uvedená hodnota 89 pre indikáciu OSPF protokolu. Jednotlivé typy správ sú popísané nižšie [12]:

- **Hello**

Hello pakety slúžia na objavovanie okolitých smerovačov pripojených na lokálnych linkách daného OSPF uzla. Slúžia na nadviazanie susedstva medzi smerovačmi a komunikáciu kľúčových parametrov. Sú posielané na multicastovú adresu 224.0.05 a obsahujú ID smerovača, ktorý pakety odoslal. V broadcastových sieťach sú Hello pakety posielané každých 10 s.

- **Database Description**

Tieto správy obsahujú popis topológie OSPF domény, teda prenášajú obsah Link State databáz medzi smerovačmi.

- **Link State Request**

Správy slúžiace na vyžiadanie si konkrétnych informácií z Link State databázy iného smerovača. Presne špecifikujú siete, pre ktoré žiadajú detailné smerovacie informácie.

- **Link State Update**

Tieto správy sú odpoveďou na Link State Requesty. Ich obsahom sú detailné informácie z LSDB databázy určené pre smerovač, ktorý si ich vyžiadal.

- **Link State Acknowledgement**

Správy slúžiace na overenie správneho doručenia ostatných OSPF správ. Explicitne potvrdzujú doručenie Link State Update paketov.

8.4 Typy správ Link State Advertisements

Každý Link State Update paket nesie tzv. Link State Advertisement (LSA). Základné typy LSA správ sú popísané nižšie [13]:

- **LSA typu 1 (Router LSA)**

Správa generovaná smerovačmi v danej oblasti na popísanie ich priamo pripojených liniek. Slúžia na prenos tzv. Intra-area smerovacích informácií. Tieto správy sa nešíria za hranice danej OSPF oblasti.

- **LSA typu 2 (Network LSA)**

Správa generovaná Designated Routrom (DR) broadcastovaj alebo NBMA siete na popis susedných smerovačov pripojených na danom segmente. Tieto správy sa nešíria za hranice danej OSPF oblasti.

- **LSA typu 3 (Summary LSA)**

Správa generovaná smerovačom na hranici OSPF oblastí na popis smerovacích informácií susedom mimo danej oblasti (ide o tzv. Inter-area routes).

- **LSA typu 4 (Summary LSA)**

Správa generovaná smerovačom na hranici OSPF oblastí na popis smerovacích informácií o tzv. ASBR (Autonomous System Boundary Router) smerovačoch pre susedov mimo danej OSPF oblasti.

- **LSA typu 5 (External LSA)**

Správa generovaná ASBR smerovačmi na popis smerovacích informácií redistribuovaných do OSPF oblasti.

8.5 Vytváranie susedstiev medzi OSPF smerovačmi

Ako už bolo spomenuté, susedstvo smerovačov nepostačuje na výmenu správ LSA. Preto smerovače musia nadviazať priateľstvo. Počas procesu nadväzovania priateľstva prechádzajú niekoľkými stavmi, ktorých význam je popísaný nižšie [14]:

- **Status Init**

V tomto stave si smerovače vymieňajú Hello pakety a vytvárajú susedstvo. Stav indikuje, že smerovač príjíma Hello pakety svojho suseda, no zatiaľ nebola vytvorená obojsmerná komunikácia. Tá sa vytvorí až vtedy, keď smerovače uvidia svoje vlastné ID v Hello paketoch prijatých od svojho suseda.

- **Status 2-way**

Ak smerovače vidia svoje ID v Hello paketoch svojho suseda, prejdu do stavu 2-way. V tomto bode sa smerovače rozhodnú, či uzatvoria priateľstvo. Na broadcastovej a NBMA sieti prejde smerovač do stavu Full iba so susedom zvaným Designated router (DR) a Backup Designated Router (BDR). S ostatnými susedmi zostane v stave 2-way. Na point-to-point a point-to-multipoint sieťach prejde smerovač do stavu Full so všetkými susedmi pripojenými na týchto linkách.

Poznámka: Prijatie paketu Database Descriptor (DBD) od suseda v Init stave taktiež spôsobí prechod do stavu 2-way.

- **Status Exstart**

Po zvolení DR a BDR ostatné smerovače vytvoria tzv. master-slave vzťah s DR a BDR, pričom master je smerovač s vyšším ID. Je dôležité poznamenať, že DR a BDR nemusia byť vždy v role mastera (aj keď logicky očakávame, že budú mať vyššie ID). Ich voľba za DR a BDR môže byť ovplyvnená aj manuálne nastavenou prioritou a ich ID môže byť pritom nižšie. Vo vzťahu master-slave potom hrajú rolu podriadeného slave.

- **Status Exchange**

V stave Exchange dochádza k výmene paketov Database Descriptor (DBD). Tie obsahujú iba LSA hlavičky a popis obsahu celej Link State databázy. Každý DBD paket má poradové číslo, ktoré môže pri prenose inkrementovať iba master. V tomto stave smerovače posielajú aj pakety Link State Request a Link State Update, ktoré obsahujú kompletnú LSA správu. Prijaté informácie sú porovnané s aktuálnou Link State databázou smerovača a ak sú v nej dostupné novšie informácie, databáza smerovača je podľa nich aktualizovaná.

- **Status Loading**

V tomto stave prebieha samotná výmena Link State databáz. Na základe súhrnných informácií z DBD paketov smerovače posielajú LSA Requesty a žiadajú v nich konkrétne informácie z Link State databázy spriateleného smerovača. DR a BDR na ne odpovedajú paketmi LSA Update. Tieto pakety sú prijímateľom potvrdzované (acknowledged).

- **Status Full**

Keď sú smerovače v stave Full, sú plne spriatelené a ich LSDB databázy sú zosynchronizované. Ide o bežný stav OSPF smerovačov. V prípade, že zostali zaseknuté v inom stave, znamená to problém s vytváraním priateľstiev. Výnimkou je stav 2-way, ktorý je bežný na broadcastových a NBMA sieťach. V prípade týchto sietí smerovač vytvára Full priateľstvá iba s DR a BDR smerovačmi. Ostatné smerovače vidí v stave 2-way.

- **Status Down**

V stave Down bolo priateľstvo so susedným smerovačom prerušené (napr. kvôli nedoručeniu Hello paketov).

9 Špecifikácia a návrh riešenia

Cieľom nášho tímového projektu je oboznámiť sa s problematikou modelovania systémov pomocou Petriho sietí. Rozhodli sme sa modelovať správanie protokolu OSPF pomocou farebných Petriho sietí. Fungovanie protokolu OSPF je bližšie popísané v kapitole 8. V nasledujúcej kapitole pojednávame o vhodných možnostiach simulácie vybraných častí a mechanizmov, ktoré OSPF protokol používa (napr. jeho správanie pri výpadku linky, formovanie susedských vzťahov medzi smerovačmi a pod.). Výsledkom kapitoly by mali byť špecifikované požiadavky, ktoré by mal náš model spĺňať.

Pre dosiahnutie uvedeného cieľa boli potrebné nasledujúce kroky:

- oboznámiť sa so základnou terminológiou z oblasti modelovania a simulácie systémov s dôrazom na oblasť farebných Petriho sietí,
- vykonať analýzu potrebných oblastí sieťového protokolu OSPF.

Kroky, ktoré budú potrebné pre splnenie zadania:

- analyzovať a osvojiť si prácu so simulačným prostredím pre implementáciu farebnej Petriho siete,
- aplikovať koncepcie a techniky modelovania systémov použitím farebných Petriho sietí,
- navrhnúť pilotný simulačný model protokolu OSPF pomocou nadobudnutých poznatkov z analýzy prístupov k modelovaniu pomocou CPN,
- vytvoriť funkčný simulačný model použitím simulačného programového prostredia,
- analyzovať výsledky simulácie modelu.

9.1 Simulácia vybraných mechanizmov protokolu OSPF

9.1.1 Čo chceme simulovať

Našu prácu by sme chceli zamerať na modelovanie dvoch hlavných štádií konvergencie siete s implementovaným OSPF protokolom, a to vznik susedstiev medzi OSPF smerovačmi a výmena DDP (Database Description Packets). Hlavným cieľom je odsimulovať stavy, cez ktoré smerovače prechádzajú počas formovania susedstiev. Keďže prevádzka protokolu OSPF pozostáva z množstva procesov a udalostí, budeme si spočiatku všímať a modelovať len tie najzákladnejšie z nich. Počas simulácie budeme sledovať zložitosť modelu a postupne pridávať nové súčasti. Čiastkové činnosti OSPF protokolu sa pokúsime detailnejšie rozobrať na samostatných modeloch.

Tvorba modelu je veľmi dobrý spôsob, ako analyzovať správanie sa navrhovaného protokolu ešte pred jeho samotnou implementáciou, prípadne odhaliť chybové stavy, ktoré môžu nastať počas jeho prevádzky v budúcnosti. Modelovanie je taktiež užitočné pri detailnom oboznámovaní sa s činnosťou modelovaného protokolu.

9.1.2 Ako chceme simulovať?

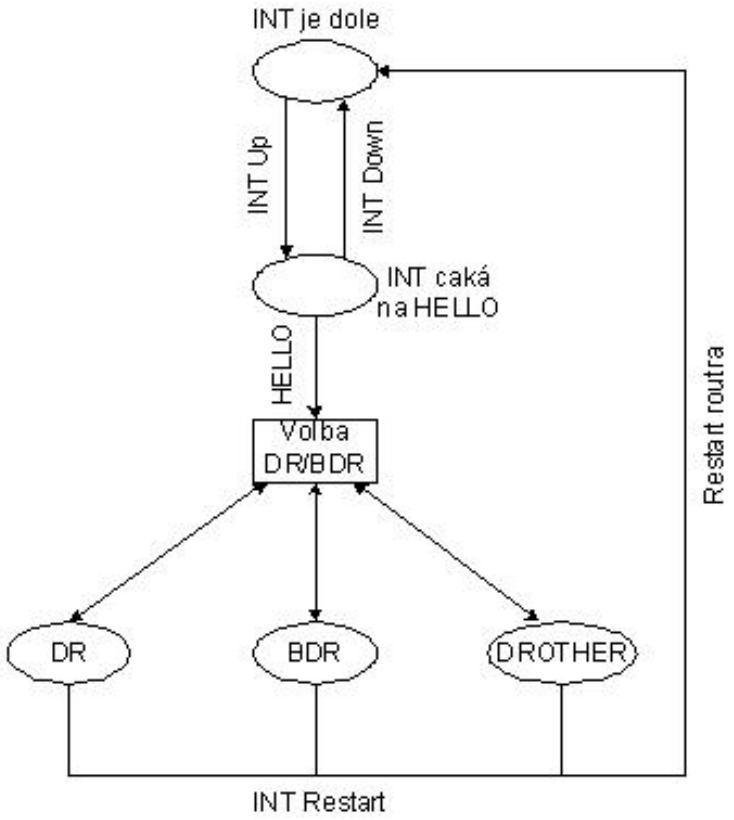
Ako typ modelu sme si zvolili Petriho siete, nakoľko sú vhodným matematickým nástrojom pre modelovanie a analýzu komplexných distribuovaných systémov so synchronizáciou udalostí. Keďže prevádzka OSPF protokolu v počítačovej sieti pozostáva z množstva prelínajúcich sa dejov, rozhodli sme sa bližšie zamerať na farebné Petriho siete. Tie boli vytvorené najmä za účelom modelovania zložitých systémov pozostávajúcich z veľkého množstva komunikujúcich procesov. Tento typ Petriho sietí sme preto zvolili za najvhodnejši spôsob simulácie prevádzky OSPF protokolu.

Ako simulačné prostredie sme si vybrali nástroj Coloured Petri Net Editor (CPN, pozri kapitolu 7.1.1).

9.2 Koncepcia modelu

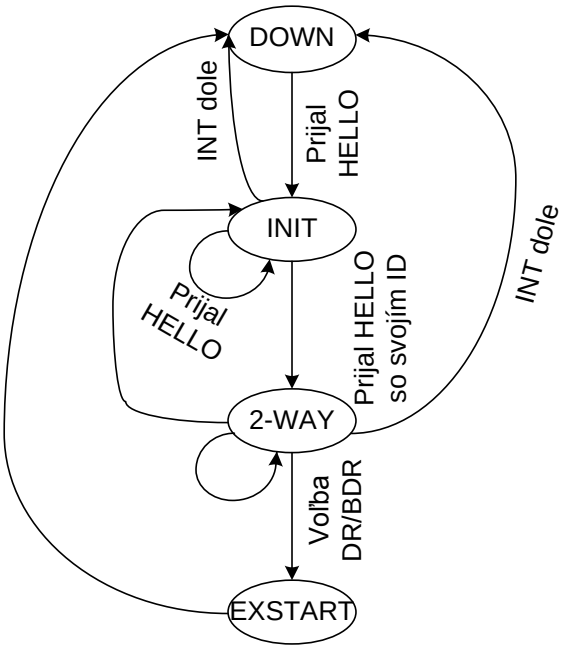
Nasledujúca kapitola popisuje základný návrh modelu protokolu OSPF. Pomocou stavových diagramov je opísaná konceptuálna funkcionalita požadovaných vlastností modelu a základná charakteristika jeho správania.

Počiatočné správanie sa protokolu OSPF je znázornené na obrázku č. 5. Obrázok znázorňuje nadviazanie susedstva medzi smerovačmi. Na začiatku je rozhranie smerovača vo vypnutom stave – stav „INT je dole“ (down state). V tomto stave si susedia nevymieňajú žiadne informácie. Po zapnutí rozhrania smerovač okrem vysielania vlastných paketov typu Hello čaká na Hello pakety od susedov. Po obdržaní a výmene Hello paketov medzi smerovačom a jeho susedmi smerovač iniciuje proces voľby DR a BDR.

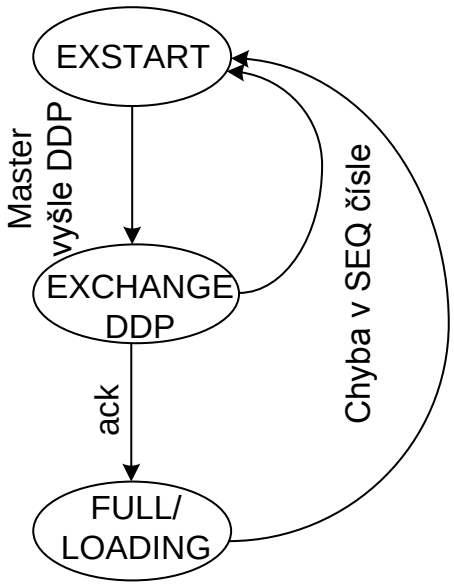


Obr. č.5: Stavový diagram procesu nadviazania spojenia a voľby DR, BDR.

Celkový model sme sa kvôli komplexnosti rozhodli rozdeliť na dva jednoduchšie moduly – modul inicializačnej fázy a modul výmeny DDP (pozri obrázky č. 6 a 7).



Obr. č.6: Modul inicializačnej fázy.



Obr.č.7: Modul Database Description Paketov

Modul inicializačnej fázy zobrazuje postupne stavy modelu smerovača vo fáze inicializácie. Nachádzame sa v prvom stave DOWN, čo znamená že rozhrania smerovača sú vypnuté a zatiaľ nebola získaná žiadna informácia. Po zapnutí rozhrania a prijatí HELLO paketu sa model dostane do stavu INIT. V tomto stave ale ešte nebola nadviazaná obojstranná komunikácia. Následne po prijatí HELLO paketu so svojím ID od susedného smerovača modul prechádza do stavu 2-WAY. Na konci tejto fázy po vzájomnej výmene informácií týkajúcich sa ID smerovačov je vykonaná voľba DR a BDR na základe typu siete (P2P alebo Broadcast/NBMA) a rozhodne sa o pokračovaní vytvárania susedstva. Model sa po voľbe DR/BDR dostane do stavu EXSTART. Smerovače sú pripravené na výmenu smerovacích informácií.

Počiatočný stav modulu DDP je EXSTART za predpokladu, že stav EXSTART je spustený iba medzi smerovačmi DROther a DR/BDR. Smerovače DR/BDR prechadzajú do stavu EXSTART za každých okolností. Cieľom stavu EXSTART je analyzovať vzťah master/slave medzi dvomi susednými smerovačmi. Rozhoduje sa na základe vyššej ID smerovača. Master vyšle ako prvý paket popisujúci smerovaciu databázu (DDP). Náš model teraz prejde do stavu EXCHANGE. Smerovače pošlú informáciu o svojej celej link-state database cez pakety DDP. Ak úspešne prebehne výmena smerovacích informácií, model sa dostáva do posledného stavu a to LOADING/FULL.

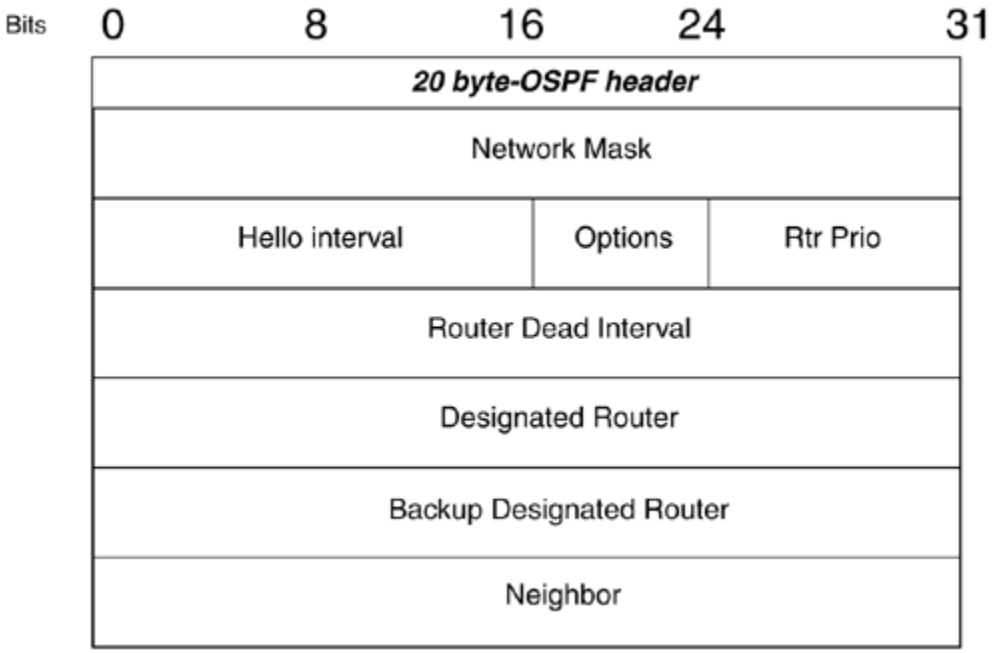
Stav LOADING umiestni každú informáciu, ktorá sa zdá byť nekompletná na link-state request zoznam. Každá vyslaná aktualizácia sa zaradí do zoznamu retransmission list, kde ostáva

do potvrdenia doručenia. Stav FULL nastane, keď link-state request list neobsahuje žiadne informácie. Je určená priľahlosť, susedné smerovače sú úplne priľahlé a majú podobné databázy. Ak nastala chyba v sekvenčnom čísle (SEQ number), tak sa model vráti do stavu EXSTART. Túto fázu sme sa však kvôli jej celkovej zložitosti požadovaného modulu rozhodli zjednodušiť a spojiť do fázy LOADING/FULL.

9.3 Prototyp modelu

Ako už bolo spomínané v predchádzajúcich kapitolách, činnosť protokolu OSPF pozostáva z procesov a stavov, ktorými prechádzajú všetky smerovače v OSPF doméne. V tejto kapitole sa budeme venovať ich modelovaniu pomocou CPN.

Pre zjednodušenie výsledného modelu tak komplexného protokolu, akým je OSPF, sme sa rozhodli rozdeliť ho na čiastkové moduly. Tieto moduly sme sa kvôli prehľadnosti rozhodli rozčleniť na jednotlivé stavy alebo procesy. Modely týchto procesov si teraz opíšeme. Predpokladáme pritom, že každý OSPF smerovač si uchováva údaje o susedných smerovačoch v definovanej dátovej štruktúre. Tá bude obsahovať vybrané položky známe z hlavičiek OSPF paketov (pozri obrázok č.8), ako napr. Router ID, DR ID, BDR ID, NEIGHBOR ID susedných smerovačov, aktuálny status smerovača, príznaky (tzv. flags) a iné.



Obr. č.8: Polia v hlavičke Hello paketu.

V CPN sme si tieto štruktúry definovali takto:

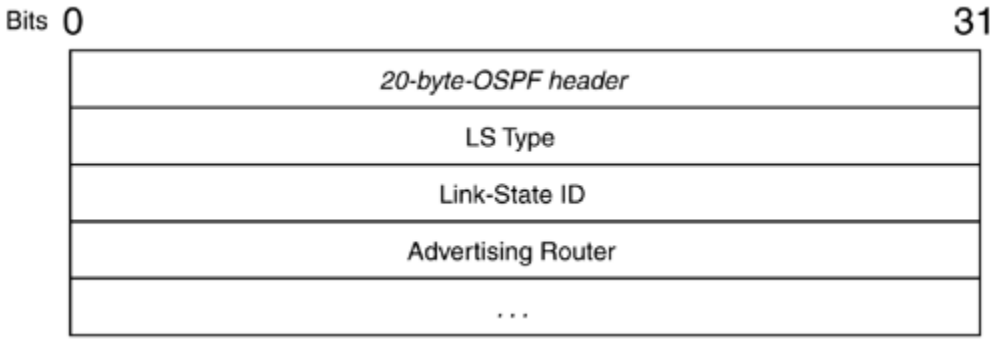
color StavRoutra = record
*status: STATUS **
*RID: ID **
*DRID: ID **
*BDRID: ID **
*mBit: FLAG **

color HelloPkt = record
*RID: ID **

Štruktúra *StavRoutra* je použitá na zaznamenanie stavu smerovača. Jeho ID je uchovávané v premennej *RID* a identifikátory jeho DR a BDR sú zaznamenané v premenných *DRID* a *BDRID*. Smerovač prechádza jednotlivými OSPF stavmi, pričom aktuálny z nich je zaznamenaý v premennej *status*. Štruktúra *HelloPkt* nesie informácie o odoslanom Hello pakete. V pokračovaní našej práce tieto štruktúry doplníme o ďalšie premenné, ktoré budeme využívať a model urobíme detailnejším. Zatiaľ predpokladáme, že doručovanie OSPF správ prenášaných IP protokolom je spoľahlivé a bez strát.

Ďalšou štruktúrou je *DDPkt*, ktorá prenáša informácie o smerovaní v OSPF doméne. Táto štruktúra je zatiaľ zjednodušená vzhľadom na skutočnú hlavičku DDP paketu kvôli komplexnosti výmeny rôznych typov LSA paketov.

color DDPkt = record
*RID: ID **
*ARID: ID * (advertising RID)*



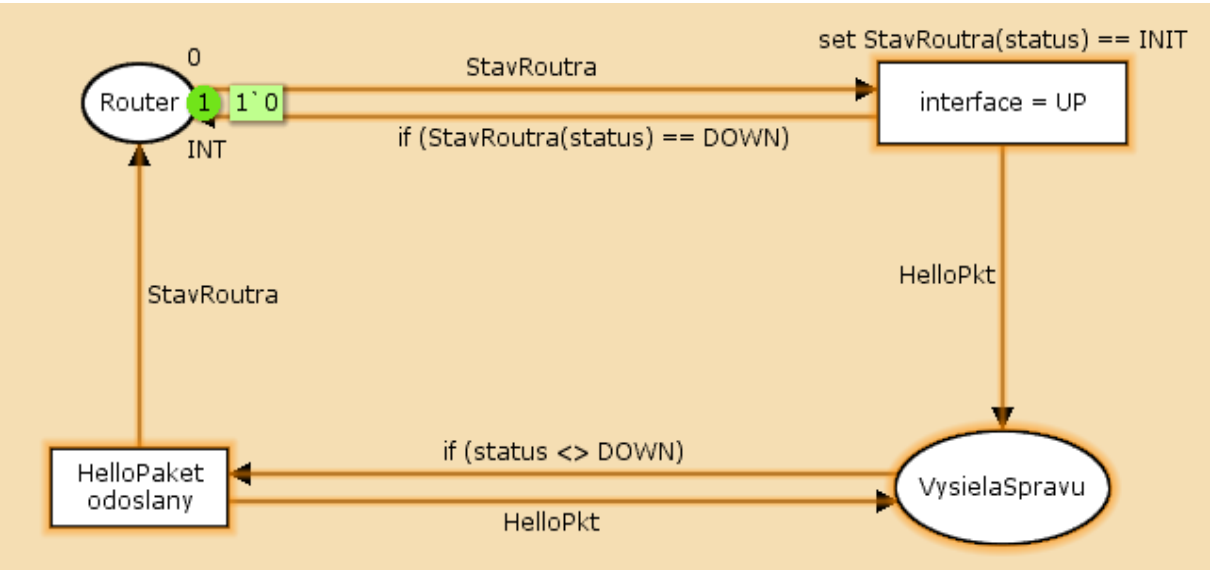
Obr. č.9: Polia hlavičky LSA paketu typu LSU.

9.3.1 Prechod zo stavu Down do stavu Init

Počiatočný stav OSPF smerovačov je Down. Počiatočné značkovanie modelu je v tomto stave nasledujúce:

1'StavRoutra({status=Dole, RID=10.0.0.1, DRID=Neznamy, DRID=Neznamy, BDRID=Neznamy, mBit=0}).

V tomto stave smerovač nevysiela a ani neprijíma pakety. V okamihu, keď jeho sieťové rozhranie príde do stavu UP (fyzicky pripojíme dátový kábel, alebo konfiguračne aktivujeme sieťové rozhranie), smerovač začne odosielať Hello pakety v pravidelných intervaloch a taktiež je schopný prijímať pakety od svojich susedov. Proces prechodu smerovača zo stavu Down do stavu Init je znázornený na obrázku č.9. Po spustení prechodu „*interface = UP*“ sa aktivuje rozhranie a status smerovača sa nastaví na INIT. Tým sa spustí proces vysielania vlastných Hello paketov za účelom oznamovania svojej prítomnosti v OSPF doméne (stav Vysiela spravu). Tým pádom je následne možné aktivovať prechod *HelloPaket odoslany*. Pri jeho spustení ale musí byť splnená podmienka, že sa status smerovača medzičasom nesmie vrátiť do stavu Down.

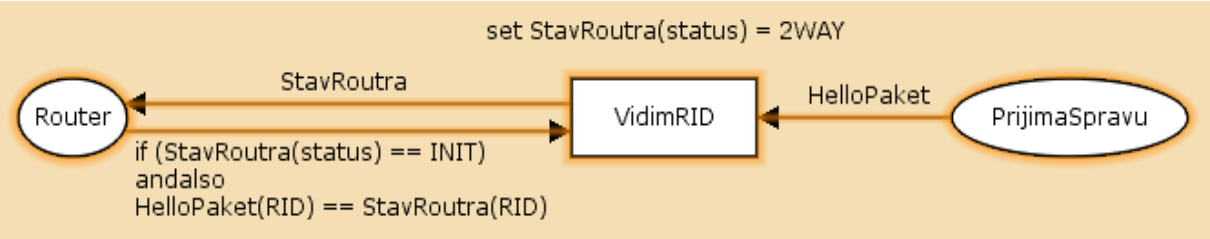


Obr. č.10: Model prechodu smerovača do stavu Init.

9.3.2 Prechod zo stavu Init do stavu 2Way

Na obrázku č.10 je znázornené prijatie Hello paketu od suseda, pričom sa skontroluje, či sa v prijatom Hello pakete nachádza ID prijímateľa (nášho smerovača). V prípade, že je splnená táto podmienka a smerovač je v stave INIT, prejde do stavu 2WAY. Tento stav značí, že výmena

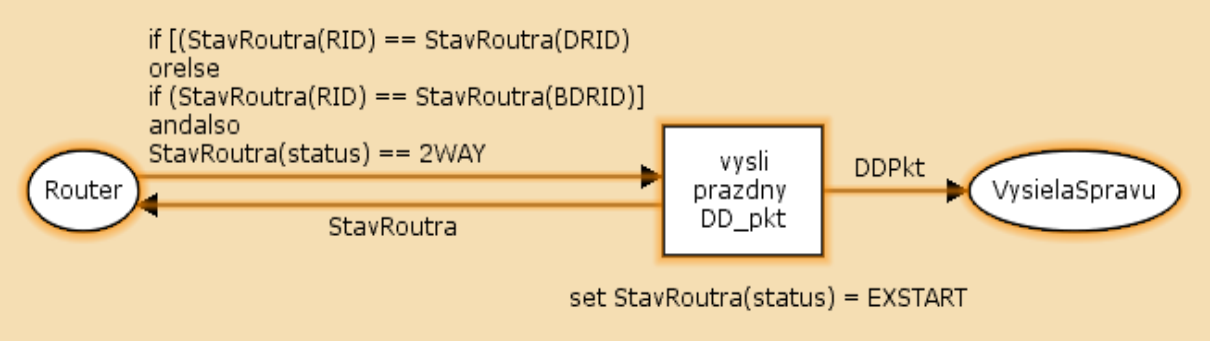
Hello paketov medzi smerovačmi je obojstranná a spätne potvrdená prítomnosťou ID prijímateľa v prijatom pakete.



Obr. č.11: Model prechodu smerovača do stavu 2 Way.

9.3.3 Prechod zo stavu 2 Way do stavu Exstart

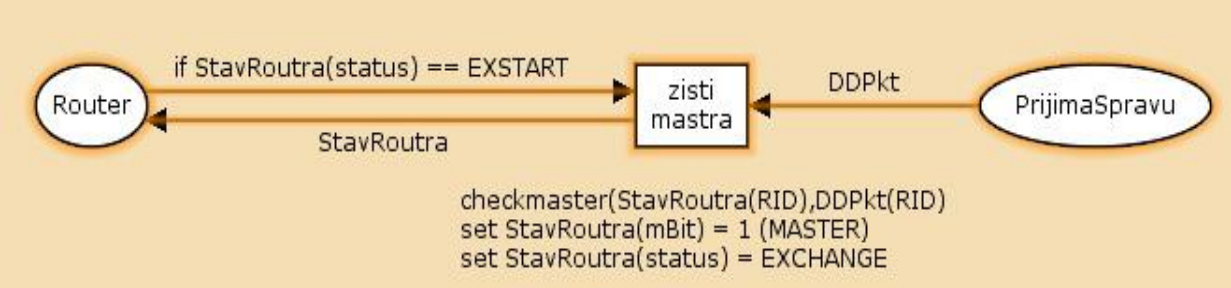
Prechod do stavu Extart je možný len s tými smerovačmi, ktoré sú zvolené za DR a BDR. S ostatnými smerovačmi zostáva opisovaný smerovač v stave 2Way. Tento proces je znázornený na obrázku č.11.



Obr.č.12: Model prechodu smerovača do stavu Exstart.

9.3.4 Prechod zo stavu Exstart do stavu Exchange

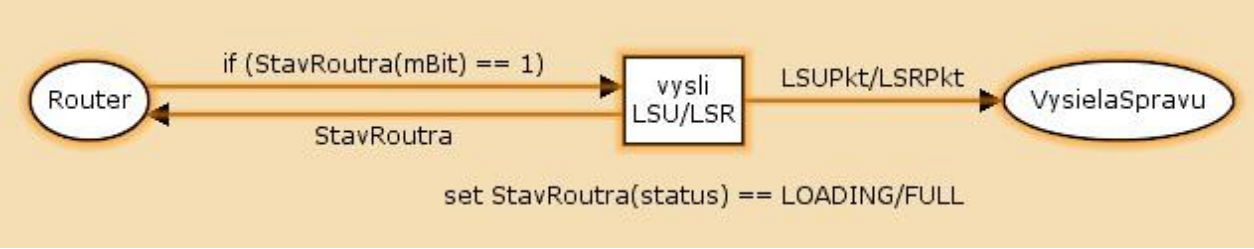
Na obrázku č.12 je znázornený proces prechodu smerovača do stavu Exchange. Tento proces zahŕňa aj voľbu tzv. Master a Slave roly pre smerovače, ktoré si idú vymieňať informácie. Smerovač s vyššou IP adresou bude Master, čo zabezpečíme nastavením príznaku mBit. Prechod do stavu Exchange znamená výmenu Database Description (DD) paketov.



Obr. č.13: Model prechodu smerovača do stavu Exchange.

9.3.5 Prechod zo stavu Exchange do stavu Loading/Full

Proces výmeny DD paketov obsahuje aj kontrolný mechanizmus, či je smerovač, ktorý vysiela správy v role Master. Tento model je zatiaľ interpretovaný zjednodušene a na základe dôkladnejšej analýzy tohoto stavu sa rozhodneme o následnom detailnejšom rozpracovaní modelovania prechodu smerovača do stavu Loading/Full, vzhľadom na veľký počet rôznych typov vyžiadaných a odoslaných Link State Request/Update paketov (LSU/LSR).



Obr. č.14: Model prechodu smerovača do stavu Loading/Full.

10 Plán ďalšej práce

V druhom semestri bude naša práca sustredená na nástroj CPN tools, ktorý využijeme pre kompletnú implementáciu a simuláciu nášho modelu. Kedže CPN tools je vcelku komplexný nástroj, bude v prvom rade potrebné oboznámiť sa so základnými dátovými štruktúrami a funkciami, ktoré poskytuje. Po hlbšom oboznámení sa s nástrojom, našu prácu preorientujeme na finalizáciu funkčného modelu OSPF protokolu, pričom využijeme nadobudnuté vedomosti z analýzy a taktiež prototyp, ktorý sme rozpracovali v zimnom semestri. Navrhnutý model naimplementujeme v spomínanom nástroji a doplníme ho o chybové stavy siete a taktiež o časovú zložku fungovania. Posledná časť práce na tímovom projekte bude venovaná simulácií modelu a analýze výsledkov.

V prípade objavenia chýb v procese implementácie bude taktiež potrebné venovať sa patričným funkcionálnym zmenám a opravám navrhnutého modelu.

11 Použitá literatúra

[1] ADAMUŠČÍNOVÁ, I.: Koncepty modelov distribuovaných systémov. Diplomová práca. Košice: FEI TU v Košiciach, 2007

[2] RADA, T.: Diagnostika porúch založená na modeloch (automatoch). Diplomová práca, FIIT STU , 2009

[3] DORČÁK, Ľ.: Matematické modelovanie procesov a systémov.

[4] Zelenka J., Matejka T.: Aplikácia teórie diskrétnych udalostných systémov na analyze a modelovanie reálneho výrobného system, 2010

[5] VICEN J., Typové príklady a využitie Petriho sietí: Bakalárska práca, vedúci BP – Ing. Tomalová, http://moodle.fiit.stuba.sk/moodle/course/view.php?id=55

[6] Žáry, M.: Prehľad techník lokalizácie porúch v počítačových sieťach, 2010

[7] Informace pro studenty předmětů PES pro léto 2009/2010. http://147.229.9.23/study/courses/PES/public/index.html.en#Aktuality

[8] Hollý, J., STOCHASTICKÉ PETRIHO SIETE II, STU Bratislava, 2009.

[9] Petri Nets Tools and software http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/

[10] OSPF Overview, History, Standards and Versions, (5.11.2010) http://www.tcpipguide.com/free/t_OSPFOverviewHistoryStandardsandVersions.htm

[11] Open Shortest Path First, (5.11.2010) http://www.cs.vsb.cz/grygarek/SPS/lect/OSPF/ospf.html

[12] OSPF General Operation and Message Types, (5.11.2010) http://www.tcpipguide.com/free/t_OSPFGeneralOperationandMessageTypes.htm

[13] CCIE Talk: OSPF LSA Types, (6.11.2010) http://www.ccietalk.com/2008/07/13/ospf-lsa-types

[14] Cisco IP Routing – OSPF Neighbor States, (5.11.2010) http://www.cisco.com/en/US/tech/tk365/technologies_tech_note09186a0080094050.shtml