# KIV/VSP – Semestrální práce

**Martin Šimek (A06101)**

Narozen 4. 1. 1984

msimek@students.zcu.cz

# Zadání

Vymyslete si otevřenou síť front (tj. propojení, vstupní proudy) obsahující alespoň 4 obslužné uzly (jeden kanál, fronta FIFO, neomez. délka), alespoň 2 vstupní proudy požadavků, alespoň 2 vnitřní zpětné vazby.

Parametry sítě (tj. střední frekvence vstupních proudů, střední doby obsluhy v jednotlivých kanálech a p-ti větvení) zvolte tak, aby síť pracovala ve stacionárním režimu. Doporučená hodnota zatížení pro všechny uzly: $\rho > 0.5$.

Určete výpočtem střední frekvence toků v uzlech. Dále určete veličiny $Lq_i$ a $Tq_i$ pro jednotlivé uzly a Lq a Tq pro celou síť pro případ, že všechny vstupní toky jsou Poissonovské a doby obsluhy ve všech uzlech mají exponenciální rozdělení.

Vypočtené hodnoty ověřte vlastnoručně vytvořeným simulačním programem. Použijte simulační knihovnu C-Sim nebo J-Sim.

Dále uvažujte případ, kdy všechny náhodné časové intervaly v modelu (příchody, obsluhy) mají Gaussovské pravděpodobnostní rozdělení N(a, σ) s (různou) střední hodnotou zvolenou v bodě 2. Vytvořte generátor tohoto rozdělení jako funkci v jazyce C nebo Java (s parametry např.a, σ) a testováním ověřte správnou funkci generátoru - chce se tedy po Vás vytvoření a prokázán správné funkce generátoru, který napíšete VY - využít můžete pouze knihovní funkce pro generování rovnoměrného rozdělení (jako v průběžném příkladu č. 1). Použití knihovní funkce pro Gaussovo rozdělení z J-SIMu se nepočítá!

Simulací ověřte chování sítě (tj. určete stejné veličiny jako v bodech 3) a 4) pro případ, že všechna rozdělení (příchody, obsluhy) budou mít hustotu N(a, σ) se stejnou střední hodnotou jako pro exponenciální rozdělení alespoň pro 3 různé hodnoty koeficientu variace C=?/a. Pokuste se o poměrně odlišné koeficienty, ať je vidět rozdíl v chování systému (např. 0.05 - tj. skoro konstantní generátor, 0.2 a 0.7 - ale to je pouze příklad). Poznámka: Simulační program je stejný jako v bodě 4, ale volá se jiný generátor podle bodu 5.

Simulační program upravte pro sledování další individuálně zadaných výkonnostních charakteristik sítě.

Řešení zpracujte formou písemného referátu (cca 10 stran, grafy, tabulky, barevné obrázky, hudební vložky, multimédia ap. - berte to jako přípravu na diplomku, navíc vlastní referát lze využít při zkoušce).

hlavní program bude možné spustit s následujícím formátem parametrů (argumentů programu):

program (počet požadavků) (EXP / GAUSS)

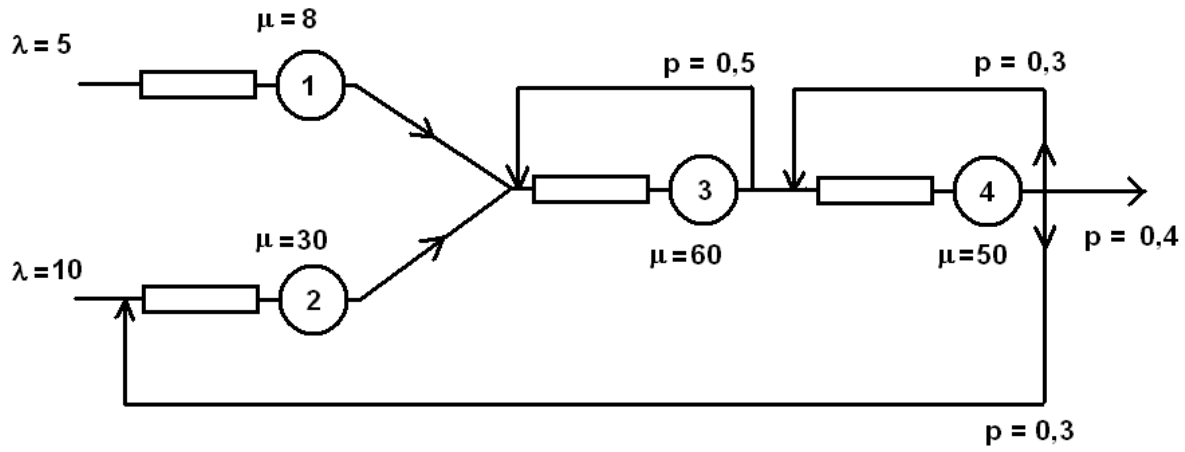
Počet požadavků udává, kolik požadavků se musí nechat simulací projít, než bude ukončena a vypočítány výsledky.

Druhým parametrem bude jeden z řetězců "EXP" nebo "GAUSS" (na velikosti písmen nechť nezáleží). V případě EXP bude spuštěna simulace pro exponenciální rozdělení, v případě GAUSS budou postupně spuštěny alespoň tři simulace pro normální rozdělení s různými koeficienty variace - tyto koeficienty budou uvedeny ve výstupu programu.

Při spuštění bez parametrů se program bude chovat, jako by byl postupně spuštěn nejdříve s parametrem EXP a poté GAUSS, počet požadavků použije nějaký defaultní.

Program (respektive rozbalený ZIP archiv) musí být dodán ve stavu schopném okamžitého spuštění na jednom z těchto systémů: WindowsXP, Linux na i586. Dle charakteru operačního systému bude připraven script "run.bat" nebo "run.sh", který zajistí spuštění programu a předání parametrů (já budu tedy při testování spouštět tento script). V případě Javovských programů předpokládejte, že jsou nastavené cesty na nainstalovanou Javu 1.6 a že nejsou nastavené cesty na žádné případné knihovny, které nejsou součástí distribuce J2SE!! Pokud budete nějaké potřebovat, přidejte si je např. do vašeho .jar souboru - to se týká i knihovny jsim.jar!

# Síť front




Soustava rovni c :

$\Lambda_1 = \lambda_1$

$\Lambda_2 = \lambda_2 + p_3\Lambda_4$

$\Lambda_3 = \Lambda_{1+}\Lambda_2 + p_1\Lambda_3$

$\Lambda_4 = (1 - p_1)\Lambda_3 + p_4\Lambda_4$

Protože $\lambda_1$ = 5 tak $\Lambda_1 = 5$

$\Lambda_2 = 10 + 0{,}3\Lambda_4$

$\Lambda_3 = 5 + \Lambda_2 + 0{,}5\Lambda_3$

$\Lambda_4 = 0{,}5\Lambda_3 + 0{,}3\Lambda_4$

Řešení jsem počítal ručně a k ověření jsem použil jeden z mnoha nástrojů dostupných online : (http://wims.unice.fr/wims/en_tool~linear~linsolver.en.html)

Řešení :

$\Lambda_1 = 5$

$\Lambda_2 = 21{,}25$

$\Lambda_3 = 52{,}5$

$\Lambda_4 = 37{,}5$

Doby obsluhy : $\mathbf{Ts_i = \frac{1}{\mu_i}}$

$\mathrm{Ts}_1 = \frac{1}{8}$ $\mathrm{Ts}_2 = \frac{1}{30}$ $\mathrm{Ts}_3 = \frac{1}{60}$ $\mathrm{Ts}_4 = \frac{1}{50}$

Zatížení : $\rho_i = \mathrm{Ts}_i \, \Lambda_i$

$\rho_1 = 0{,}63$ $\rho_2 = 0{,}71$ $\rho_3 = 0{,}88$ $\rho_4 = 0{,}75$

Všechna ρ jsou menší než 1 systém je tedy stabilní.

Průměrný počet požadavků v uzlech : $\mathbf{Lq_i = \frac{\rho_i}{1-\rho_i}}$

$\mathrm{Lq}_1 = 1{,}70$ $\mathrm{Lq}_2 = 2{,}44$ $\mathrm{Lq}_3 = 7{,}33$ $\mathrm{Lq}_4 = 3$

Průměrný doba průchodu uzlem : $\mathbf{Tq_i = \frac{Ts_i}{1-\rho_i}}$

$\mathrm{Tq}_1 = 0{,}34$ $\mathrm{Tq}_2 = 0{,}11$ $\mathrm{Tq}_3 = 0{,}14$ $\mathrm{Tq}_4 = 0{,}08$

Celkový průměrný počet požadavků v uzlech : $Lq = \sum Lq_i$

$Lq = 14{,}47$

Celkový průměrná doba průchodu systémem : $Tq = \sum Tq_i$

$Tq = 0{,}67$

# Simulace

K simulaci jsem použil příklad číslo 8 na Queuing Networks z J-Simu, který jsem patřičně upravil – k jednotlivým třídám přidal proměnné potřebné k sledování požadovaných údajů. Server jsem upravil tak, aby bylo možné udělat větvení na tři různé možnosti.

Požadavky jsou reprezentovány třídou ***Transaction***. Objekty této třídy si sebou nesou čas vytvoření a seznam uzlů (spolu s časy), kterými prošly. Název uzlu a čas je sloučen v třídě ***ZastavkaPozadavku***.

Fronty jsou reprezentovány třídou ***QueueWithServer,*** která je potomkem ***JSimHead***. Fronty mají navíc počítadlo požadavků a počítadlo časů uplynulých mezi příchody požadavků.

Generátory požadavků jsou reprezentovány třídou ***Generator***, která je potomkem ***JSimProcess***. Dostanou parametr rozdělení a koeficient variace a během svého života vytvářejí požadavky a umisťují je do fronty,na kterou mají odkaz. Pokud je koeficient variace nulový, použije se při čekání mezi vytvořením požadavků náhodný generátor s exponenciálním rozdělením, jinak se použije generátor s gaussovským rozdělením.

Servery jsou reprezentovány třídou ***Server***, která je potomkem ***JSimProcess***. Oproti původní verzi mají jednu vstupní frontu a až dvě odchozí, mezi něž jsou požadavky rozesílány na základě dvou pravděpodobností předaných jako parametr.

Pro generování náhodných čísel s gaussovským rozdělením je volána statická funkce třídy ***GaussGenerator*** – *vygenerujCislo(a, sigma)* s parametry střední hodnotou a rozptylem.

Vlastní simulace je inicializována a pouštěna v třídě ***QueueingNetworkExample.*** Kde jsou vyhodnoceny parametry programu, a poté nastaveny a spuštěny příslušné simulace. Poté, co sítí proběhne zadaný počet požadavků se simulace ukončí a vyhodnotí.

Simulace v J-Sim pro 100 000 požadavků :

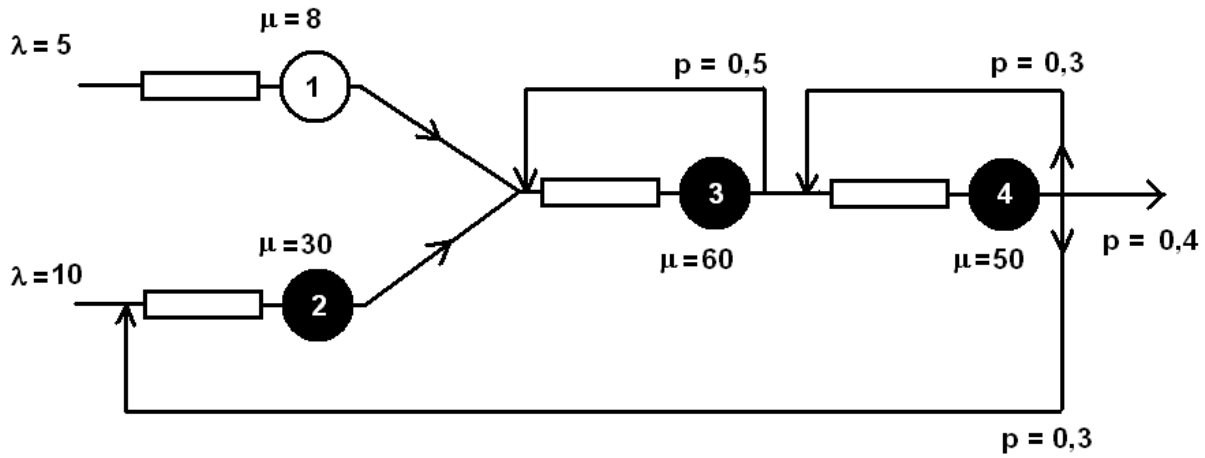
| | $Lq_1$ | $Tq_1$ | $Lq_2$ | $Tq_2$ | $Lq_3$ | $Tq_3$ | $Lq_4$ | $Tq_4$ | **Lq** | **Tq** |
|---|---|---|---|---|---|---|---|---|---|---|
| **Výpočet exponenciální rozdělení** | 1,7 | 0,34 | 2,44 | 0,11 | 7,33 | 0,14 | 3 | 0,08 | **14,5** | **0,67** |
| **Simulace exponenciálního rozdělení** | 1,62 | 0,32 | 2,44 | 0,11 | 6,83 | 0,13 | 3,22 | 0,08 | **14,1** | **0,64** |
| **Simulace Gaussovské rozdělení Cs = 0,8** | 1,24 | 0,37 | 1,96 | 0,13 | 6,04 | 0,17 | 2,72 | 0,10 | **12** | **0,77** |
| **Simulace Gaussovské rozdělení Cs = 0,5** | 0,81 | 0,17 | 1,41 | 0,07 | 4,37 | 0,08 | 2,07 | 0,05 | **8,66** | **0,37** |
| **Simulace Gaussovské rozdělení Cs = 0,2** | 0,62 | 0,12 | 1,10 | 0,05 | 3,29 | 0,06 | 1,71 | 0,04 | **6,72** | **0,27** |

# Další sledované statistiky

Dalším cílem simulací bylo vytvoření statistik pro požadavky jdoucí cestou 2-3-4 :

- kolik % požadavků touto cestou chodí
- doba průchodu cestou
- střední hodnota
- rozptyl
- histogram

Za průchod cestou jsem považoval průchod po uzlech 2-3-4 přesně v tomto pořadí, ačkoliv po této cestě teoreticky chodí i požadavky s cestou např. 2-3-3-4 , 2-3-4-4, 2-3-3-4-4 atd.



Při každém opuštění uzlu se do požadavku přidá název uzlu a čas opuštění. Při opouštění sítě pak požadavek předá seznam těchto hodnot a z nich se určuje, kolik požadavků prošlo vybranou cestou a jak dlouho jim to trvalo. Doba průchodu cestou je počítána jako rozdíl časů před vstoupením do uzlu 2 a po opuštění uzlu 4.

Seznam požadavků jdoucích přímo po cestě (tj. vstup do 2, 3, 4 a ven) : **13%**

Seznam požadavků jdoucích aspoň jednou přímo po cestě (tj. v celkové cestě požadavku se aspoň jednou objeví sekvence 2-3-4) : **51%**

Kolikrát jednotlivé požadavky prošly cestou :

- 0x **48310** tj. **48%**
- 1x **37519** tj. **37%**
- 2x **10279** tj. **10%**
- 3x **2756** tj. **2%**
- 4x **837** tj. **0%**
- 5x+ **299** tj. **0%**

Střední hodnota doby průchodu cestou : **0.28**

Rozptyl doby průchodu cestou : **0.043**

Histogram (Jedna hvězdička je cca : 124 hodnot)

```
-0,587
-0,543
-0,500
-0,457
-0,413
-0,370
-0,327
-0,283
-0,240
-0,196
-0,153
-0,110
-0,066
-0,023
0,021 ************************************
0,064 ************************************************
0,107 ************************************************************
0,151 ************************************************************
0,194 ********************************************************
0,237 ************************************************
0,281 ******************************************
0,324 ************************************
0,368 ********************************
0,411 ************************
0,454 ********************
0,498 ****************
0,541 **************
0,585 ***********
0,628 **********
0,671 ********
0,715 ******
0,758 *****
0,801 ***
0,845 **
0,888 **
0,932 *
0,975 *
1,018 *
1,062
1,105
```

Histogram má charakter Erlangova rozdělení třetího stupně, protože je produktem součtů tří náhodných veličin s exponenciálním rozdělením.

# Generátor Gaussovského rozdělení

Část testování měla být s použitím vlastnoručně vytvořeného generátoru náhodných čísel s gaussovským rozdělením. Jelikož tvorba tohoto generátoru byla zadáním mého prvního příkladu, použil jsem už hotový s prokázanou funkčností. Jedinou změnou bylo opakování generování čísla, pokud by byl výsledek záporný (doba obsluhy přirozeně nemůže být záporná).

Použitý vzorec :

$$x = a + \sigma\left(\left(\sum_{i=1}^{12} y_i\right) - 6\right)$$

```
public static double vygenerujCislo(double a, double sigma) {

            Random generator = new Random();

            double suma = 0;

            double nahodneCisloRR;
            double nahodneCisloGR;

            do {
                  for (int j = 0; j < 12; j++) {
                        nahodneCisloRR = generator.nextFloat();
                        suma += nahodneCisloRR;
                  }
                  // vzorec pro generovani gaussova rozdeleni
                  nahodneCisloGR =  sigma*(suma -6) + a;
                  //System.out.println(nahodneCisloGR);
            } while(nahodneCisloGR < 0);

      return nahodneCisloGR;
}
```

# Závěr

Hodnoty získané ze simulace se dle mého názoru dostatečně blíží k hodnotám spočítaným teoreticky. Tvorba simulace (potažmo úprava ukázkového příkladu) byla relativně snadná, na druhou stranu, některé problémy bych bez rady spolužáků řešil velmi obtížně.

Návrh sítě, do kterého mi pan Racek přidal jednu zpětnou vazbu jsem musel lehce modifikovat, resp. rozhodnout se, kterou ze dvou možných interpretací použiji. Věřím, že zvolenou možností jsem si práci nijak neulehčil.

Zpracovaná statistika cesty má všechny potřebné údaje, pouze doufám, že jsem pojem cesta pochopil správně jako průchod požadavku přes danou posloupnost uzlů, bez jakýchkoliv smyček uvnitř cesty.

Použitý generátor gaussovského rozdělení je generátor vytvořený pro mou první úlohu a nepovažoval jsem tedy za nutné znovu testovat a prezentovat jeho funkčnost, navíc vzhledem k tomu, že způsob tohoto otestování není v zadání jasně popsán.