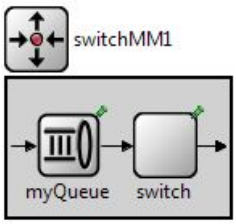
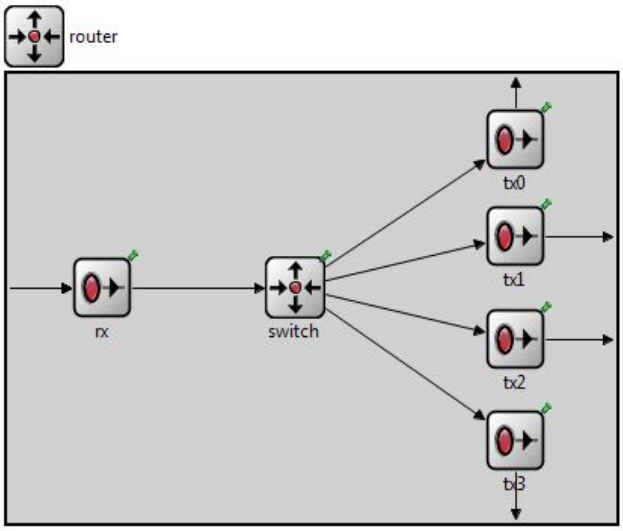
Naloga 4

1.) Predlogi za domačo nalogo je potrebno dodati enoto za usmerjanje (v podanih datotekah je že deklarirana kot switchMM1, zato bomo uporabili kar to ime). Ker moramo poleg usmerjevalnika realizirati tudi čakalno vrsto, bo switchMM1 modula, ki bo vsebovala preprosti 'switch' in pa že vključeno enoto myQueue, katero bomo 'zaklenili' na zgolj en strežnik:



Switch zgolj trivialno usmerja paket na dodeljeni naslov, kjer je le-ta v bistvu index izhodnih vrat na routerju. Verjetno bi bilo lažje čakalno vrsto realizirati kar znotraj routerja in ne v gnezdeni pod-moduli, vendar sem se pepozno zavedel, da bo 'switchMM1' modula in ne zgolj simple gradnik. Okoli switch-a izgleda podani router takole:



2.) Analiza:

a. Optimalno število strežnih enot določimo s poizkusom ogromnega števila resursov, in nato zmanjšamo število strežnikov na najmanjše naravno število, ki je večje od povprečnega števila zasedenih strežnikov. S tem poskrbimo za največjo izkoriščenost resursov, za potencialno ceno povečanja uporabe čakalnih vrst, vendar le-te ne bodo naraščale v neskončnost.

b. Povezava je še pri 500/500 bps delovala brez napak...

c. Povprečni čas med poslano zahtevo in prejetim odgovorom bomo delno izračunali, delno pa izmerili. Izmerili bomo čas, ki ga zahteva preživi med prenašanjem po mediju, ter povprečni čas, ki ga preživi v čakalnih vrstah, izračunali pa čas, ki ga preživi znotraj strežniških enot. Čase na vodilih smo izmerili:

seminarska1.server1.rx	travelTime:mean	4.48E-4
seminarska1.server2.rx	travelTime:mean	4.4799999999979E-4
seminarska1.router.rx	travelTime:mean	0.0021930239558179
seminarska1.server3.rx	travelTime:mean	4.4799999999984E-4
seminarska1.client.rx	travelTime:mean	0.0039380961007405

Časi do routerja in časi do klienta so za vse prenose enaki, zato jih lahko kar poračunamo. Za preostale čase, pa moramo upoštevati pogostost uporabe posameznega strežnika.

client.rx + router.rx + (server1.rx * 0.1 + server2.rx * 0.5 + server3.rx * 0.4) =

0.003938 + 0.002193 + (0.00048 * 0.1 + 0.000448 * 0.5 + 0.000448 * 0.4) =

0.003938 + 0.002193 + 0.0004512 = 0.0065822

Poračunamo še čase v čakalnih vrstah:

seminarska1.server1.servUnit	waitTime:mean	0.031175116624062
seminarska1.server2.servUnit	waitTime:mean	0.0015466721100631
seminarska1.router.switch.myQueue	waitTime:mean	1.4898215755333E-6
seminarska1.server3.servUnit	waitTime:mean	0.0

Čas znotraj switcha močno izstopa, tudi zato, ker je znotraj zajeto čakanje iz obeh smeri, zato ga bomo tudi računali zgolj enkrat. Ponovno je potrebno upoštevati pogostost uporabe strežnikov.

router.switch.myQueue + (server1.servUnit * 0.1 + server2.servUnit * 0.5 + server3.servUnit * 0.4)
1.489821 + (0.031175 * 0.1 + 0.001546 * 0.5 + 0) = 1.489821 + 0.003891 = 1.493712

Dodajmo še čase procesiranj:
switch.usmerjanje + (server1.strezba * 0.1 + server2.strezba * 0.5 + server3.strezba * 0.4)
0.001 + (0.5 * 0.1 + 0.1 * 0.5 + 0.05 * 0.4) = 0.001 + 0.12 = 0.121

Iz tega sledi, da je povprečni čas med zahtevo in odgovorom:
0.0065822 + 1.493712 + 0.121 = 1.6212942

Opazimo, da so največji krivec za zamude čakalne vrste, če smo natančnejši prav čakalna vrsta znotraj usmerjevalnika. Seveda, saj ima tudi štirikratno obremenitev...

d. Začuda pri prenosu ni prihajalo do napak:

Network-0-20...	seminarska1.client.tx	transmitError:count	0.0
Network-0-20...	seminarska1.client.rx	transmitError:count	0.0
Network-0-20...	seminarska1.router.rx	transmitError:count	0.0
Network-0-20...	seminarska1.router.tx0	transmitError:count	0.0
Network-0-20...	seminarska1.router.tx1	transmitError:count	0.0
Network-0-20...	seminarska1.router.tx2	transmitError:count	0.0
Network-0-20...	seminarska1.router.tx3	transmitError:count	0.0
Network-0-20...	seminarska1.server1.tx	transmitError:count	0.0
Network-0-20...	seminarska1.server1.rx	transmitError:count	0.0
Network-0-20...	seminarska1.server2.tx	transmitError:count	0.0
Network-0-20...	seminarska1.server2.rx	transmitError:count	0.0
Network-0-20...	seminarska1.server3.tx	transmitError:count	0.0
Network-0-20...	seminarska1.server3.rx	transmitError:count	0.0

Če bi prihajalo, bi iz naslova recieverja/transmitterja ugotavljali ali zaradi prepolnega kanala(tx) ali zaradi error-bita(rx).