Project 4: Switch Performance and HOL Blocking

EE 511 – Section: Tuesday 5 pm

Name: Darshan Patil

Student ID: 9575227834

1.

i. Problem Statement

As discussed in class, you are study the operation of an $N * N$ switch under heavy traffic, i.e. there are always packets on the input ports. In particular, the HOL (head of the line) slot is always full. The packet in the HOL position at any input of the N input ports is addressed to output port $j : j = 1, \ldots, N$ with probability α_j and $\sum_{j=1}^{N} \alpha_j = 1$. Packets can be delivered from inputs to outputs in one clock cycle and the clock rate is 106 cycles per second. If there is more than 1 packet destined to a specific port, only one of them can be delivered in the current slot, the others will remain in the HOL position on the input side. This will reduce switch throughput and is known as HOL-Blocking. Estimate the rate in pps (pps = packets per second) at which packets are delivered to each output port and the overall throughput of the switch (also in pps.)

$N = 2$

We are interested in finding the switch throughput for a range of values for $\alpha_1 = \{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$ and recall that $\alpha_2 = 1 - \alpha_1$.

Method 1: Build the transition probability matrix (see class notes) and solve the Markov chain numerically to find the limit distribution. From this limit distribution, calculate throughput for each output port and the overall switch performance.

Method 2: Build a simulation model that simulates the operation of the switch. In each slot, one or two packets will be delivered to the output. The packets that are transmitted are replaced in the HOL positions with new packets with destination ports generated randomly according to α_j .

For $N = 4 \; and \; N = 8$, appropriately generalize the simulation program for these cases (the numerical solution to the balance equations is not required). The number of packets delivered to any particular output in one slot is either 1 or 0. When there are multiple requests for the same output, one of them is selected to be delivered and the others remain in the HOL position on the input side.

Simulate for 1) balanced traffic $\left(\alpha_j = \frac{1}{N}\alpha \;\; for \; all \; j \; and \; 2\right)$

Hot-spot traffic $\alpha_1 = \frac{1}{k}, \alpha_j = \left(\frac{1}{N-1}\right)\left(\frac{k-1}{k}\right) \; for \; j \neq 1$. Look at the cases $k = 2, 3, \ldots, N$ (the case K=N reverts to being balanced traffic)

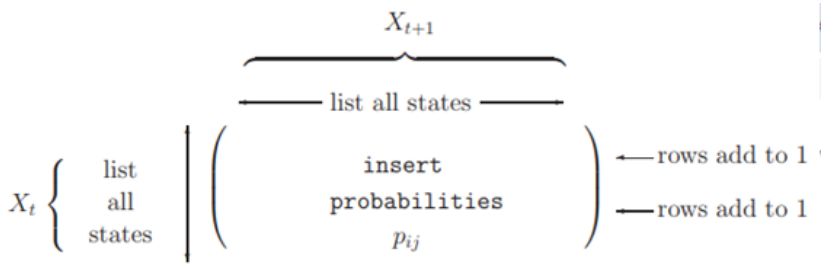
ii. Theoretical Exploration

Markov Chain

A Markov chain is "a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event". We describe a Markov chain as follows: We have a set of states, $S = \{s1, s2\ldots, sr\}$. The process starts in one of these states and moves successively from one state to another. Each move is called a step. If the chain is currently in state s_j, then it moves to state s_j at the next step with a probability denoted by p_{ij}, and this probability does not depend upon which states the chain was in before the current state. The probabilities p_{ij} are called transition probabilities. The process can remain in the state it is in, and this occurs with probability p_{ij}. An initial probability distribution, defined on S, specifies the starting state. Usually this is done by specifying a state as the starting state.

Transition Matrix

The transition matrix records all data about transitions from one state to the other.



The transition matrix is usually given the symbol $P = (P_{ij})$. In the transition matrix P: • the ROWS represent NOW, or FROM (X_t); the COLUMNS represent NEXT, or TO (X_{t+1}); entry (i, j) is the CONDITIONAL probability that NEXT = j, given that NOW = i: the probability of going FROM state i TO state j.

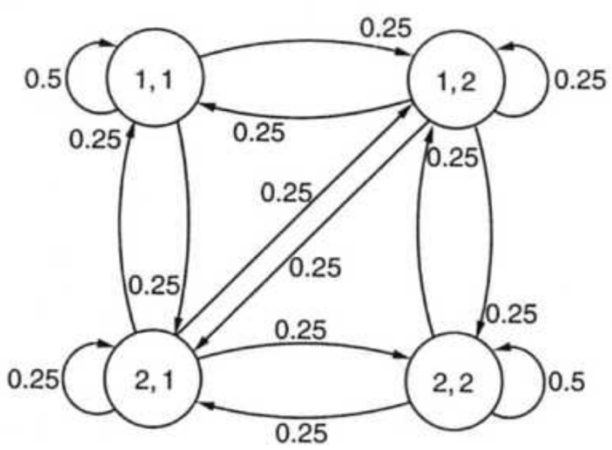
$$p_{ij} = \mathbb{P}(X_{t+1} = j \mid X_t = i).$$

Limiting distribution for a Markov chain

Under suitable easy-to-check conditions, we will see that a Markov chain possesses a limiting probability distribution, $\pi = (\pi j)\ j \in S$, and that the chain, if started off initially with such a distribution will be a stationary stochastic process and can be obtained by solving a set of linear equations. For a 2×2 matrix the value of matrix is as shown below.

$$p = \begin{bmatrix} \alpha_0 & \frac{\alpha_1}{2} & \frac{\alpha_1}{2} & 0 \\ \alpha_0^2 & \alpha_0\alpha_1 & \alpha_0\alpha_1 & \alpha_1^2 \\ \alpha_0^2 & \alpha_0\alpha_1 & \alpha_0\alpha_1 & \alpha_1^2 \\ 0 & \frac{\alpha_0}{2} & \frac{\alpha_0}{2} & \alpha_1 \end{bmatrix}$$

State diagram for 2*2 switch, $\alpha_0 = \alpha_1 = 0.5$



The transition matrix of the Markov chain is $P = (p_{ij})$. By analysing the problem, we need to generate a series of RVs which is uniformly distributed from 0 to 1. Let's say the numbers in the sequence to be 1000. Then we can have 1000-time slots. The parameters of the processing can be extracted from these RVs. Then we can iterate the whole process for 10000 times and we will find the distribution of what we are interested in.

Also, we can see that increasing the packets will lead to the increase in both of Mean of N packets in the buffer1 and Mean of N packets processed per slot. This is because the probability that a packet will arrive to input is increasing then the traffic will be heavier which apparently will lead to more conflicts but at the same time higher throughput.

iii. Results

Analytical Calculation of Transition Matrix for N=2 i.e.,2*2 switch;

$\alpha_0 = \Pr[packet\ wants\ o/p\ 0]$

$\alpha_1 = \Pr[packet\ wants\ o/p\ 1]$

$$p = \begin{bmatrix} \alpha_0 & \frac{\alpha_1}{2} & \frac{\alpha_1}{2} & 0 \\ \alpha_0^2 & \alpha_0\alpha_1 & \alpha_0\alpha_1 & \alpha_1^2 \\ \alpha_0^2 & \alpha_0\alpha_1 & \alpha_0\alpha_1 & \alpha_1^2 \\ 0 & \frac{\alpha_0}{2} & \frac{\alpha_0}{2} & \alpha_1 \end{bmatrix}$$

$Lets\ calculate\ switch\ throughput\ for\ \alpha_0 = \{0, 0.1, 0.2, 0.3, 0.4, 0.5\}\ and\ \alpha_1 = 1 - \alpha_0.$

For $\alpha_0 = 0, \alpha_1 = 1$

$$p = \begin{bmatrix} 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$\pi_{00} = \pi_{00}(0) + \pi_{01}(0) + \pi_{10}(0) + \pi_{11}(0)$
$\pi_{01} = \pi_{00}(0.5) + \pi_{01}(0) + \pi_{10}(0) + \pi_{11}(0)$
$\pi_{10} = \pi_{00}(0.5) + \pi_{01}(0) + \pi_{10}(0) + \pi_{11}(0)$
$\pi_{11} = \pi_{00}(0) + \pi_{01}(1) + \pi_{10}(1) + \pi_{11}(1)$

$\pi_{00} + \pi_{01} + \pi_{10} + \pi_{11} = 1$

Solving above equations, we get,
$\pi_{00} = 0, \pi_{10} = 0, \pi_{01} = 0, \pi_{11} = 1$
$throughput = \pi_{00}[1] + (\pi_{01} + \pi_{10}) * 2 + \pi_{11}[1] = 0 + (0 + 0) * 2 + 1 = 1\ pps$
$pps =\ packets\ per\ slot$
$Overall\ switch\ performance = \frac{1}{2} * 100 = 50\%$
$Throughput\ for\ output\ port\ 0\ =\ \pi_{00}[1] + (\pi_{01} + \pi_{10}) * 2 + \pi_{11}[1] =\ 0$
$Throughput\ for\ output\ port\ 1\ =\ \pi_{00}[1] + (\pi_{01} + \pi_{10}) * 2 + \pi_{11}[1]\ =\ 1$

For $\alpha_0 = 0.1, \alpha_1 = 0.9$

$$p = \begin{bmatrix} 0.1 & 0.45 & 0.45 & 0 \\ 0.01 & 0.09 & 0.09 & 0.81 \\ 0.01 & 0.09 & 0.09 & 0.81 \\ 0 & 0.05 & 0.05 & 0.9 \end{bmatrix}$$

$\pi_{00} = \pi_{00}(0.1) + \pi_{01}(0.01) + \pi_{10}(0.01) + \pi_{11}(0)$
$\pi_{01} = \pi_{00}(0.45) + \pi_{01}(0.09) + \pi_{10}(0.09) + \pi_{11}(0.05)$
$\pi_{10} = \pi_{00}(0.45) + \pi_{01}(0.09) + \pi_{10}(0.09) + \pi_{11}(0.05)$
$\pi_{11} = \pi_{00}(0) + \pi_{01}(0.081) + \pi_{10}(0.081) + \pi_{11}(0.9)$
$\pi_{00} + \pi_{01} + \pi_{10} + \pi_{11} = 1$

Solving above equations, we get,
$\pi_{00} = 0.0012, \pi_{10} = 0.0549, \pi_{01} = 0.05488, \pi_{11} = 0.8890$
$throughput = \pi_{00}[1] + (\pi_{01} + \pi_{10}) * 2 + \pi_{11}[1] = 0.0012 + (0.0549 + 0.05488) * 2 + 0.8890$
$= 1.1098\ pps$
$Throughput\ for\ output\ port\ 0\ =\ \pi_{00}[1] + (\pi_{01} + \pi_{10}) * 2 + \pi_{11}[1] =\ 0.0012\ +\ 0.549\ *\ 2$
$= 0.111\ pps$
$Throughput\ for\ output\ port\ 1\ =\ \pi_{00}[1] + (\pi_{01} + \pi_{10}) * 2 + \pi_{11}[1]$
$= 0.05488\ *\ 2\ +\ 0.8890\ =\ 0.99876\ pps$
$Overall\ switch\ performance = \frac{1.1098}{2} * 100 = 55.49\%$

For $\alpha_0 = 0.2, \alpha_1 = 0.8$

$$p = \begin{bmatrix} 0.2 & 0.4 & 0.4 & 0 \\ 0.04 & 0.16 & 0.16 & 0.64 \\ 0.04 & 0.16 & 0.16 & 0.64 \\ 0 & 0.1 & 0.1 & 0.8 \end{bmatrix}$$

$\pi_{00} = \pi_{00}(0.2) + \pi_{01}(0.04) + \pi_{10}(0.04) + \pi_{11}(0)$
$\pi_{01} = \pi_{00}(0.4) + \pi_{01}(0.16) + \pi_{10}(0.16) + \pi_{11}(0.1)$
$\pi_{10} = \pi_{00}(0.4) + \pi_{01}(0.16) + \pi_{10}(0.16) + \pi_{11}(0.1)$
$\pi_{11} = \pi_{00}(0) + \pi_{01}(0.64) + \pi_{10}(0.64) + \pi_{11}(0.8)$
$\pi_{00} + \pi_{01} + \pi_{10} + \pi_{11} = 1$
Solving above equations, we get,
$\pi_{00} = 0.01176, \pi_{10} = 0.117647, \pi_{01} = 0.117647, \pi_{11} = 0.75294$

$$throughput = \pi_{00}[1] + (\pi_{01} + \pi_{10}) * 2 + \pi_{11}[1]$$
$$= 0.01176 + (0.1176 + 0.1176) * 2 + 0.75294 = 1.2351\ pps$$
$$Throughput\ for\ output\ port\ 0 = \pi_{00}[1] + (\pi_{01} + \pi_{10}) * 2 + \pi_{11}[1]$$
$$= 0.01176 + 0.1176 * 2 = 0.24696$$
$$Throughput\ for\ output\ port\ 1 = \pi_{00}[1] + (\pi_{01} + \pi_{10}) * 2 + \pi_{11}[1]$$
$$= 0.117647 * 2 + 0.75294 = 0.98823$$

$$Overall\ switch\ performance = \frac{1.2351}{2} * 100 = 61.755\%$$

For $\alpha_0 = 0.3, \alpha_1 = 0.7$

$$p = \begin{bmatrix} 0.3 & 0.35 & 0.35 & 0 \\ 0.09 & 0.21 & 0.21 & 0.49 \\ 0.09 & 0.21 & 0.21 & 0.49 \\ 0 & 0.15 & 0.15 & 0.7 \end{bmatrix}$$

$\pi_{00} = \pi_{00}(0.3) + \pi_{01}(0.09) + \pi_{10}(0.09) + \pi_{11}(0)$
$\pi_{01} = \pi_{00}(0.35) + \pi_{01}(0.21) + \pi_{10}(0.21) + \pi_{11}(0.15)$
$\pi_{10} = \pi_{00}(0.35) + \pi_{01}(0.21) + \pi_{10}(0.21) + \pi_{11}(0.15)$
$\pi_{11} = \pi_{00}(0) + \pi_{01}(0.49) + \pi_{10}(0.49) + \pi_{11}(0.7)$
$\pi_{00} + \pi_{01} + \pi_{10} + \pi_{11} = 1$
Solving above equations, we get,
$\pi_{00} = 0.0465, \pi_{10} = 0.181, \pi_{01} = 0.181, \pi_{11} = 0.5913$

$$throughput = \pi_{00}[1] + (\pi_{01} + \pi_{10}) * 2 + \pi_{11}[1] = 0.0465 + (0.181 + 0.181) * 2 + 05913$$
$$= 1.3619\ pps$$
$$Throughput\ for\ output\ port\ 0 = \pi_{00}[1] + (\pi_{01} + \pi_{10}) * 2 + \pi_{11}[1]$$
$$= 0.0465 + 0.1810 * 2 = 0.4085$$
$$Throughput\ for\ output\ port\ 1 = \pi_{00}[1] + (\pi_{01} + \pi_{10}) * 2 + \pi_{11}[1]$$
$$= 0.1810 * 2 + 0.5913 = 0.9533$$
$$Overall\ switch\ performance = \frac{1.3619}{2} * 100 = 68.09\%$$

For $\alpha_0 = 0.4, \alpha_1 = 0.6$

$$p = \begin{bmatrix} 0.4 & 0.3 & 0.3 & 0 \\ 0.16 & 0.24 & 0.24 & 0.36 \\ 0.16 & 0.24 & 0.24 & 0.36 \\ 0 & 0.2 & 0.2 & 0.6 \end{bmatrix}$$

$\pi_{00} = \pi_{00}(0.4) + \pi_{01}(0.16) + \pi_{10}(0.16) + \pi_{11}(0)$
$\pi_{01} = \pi_{00}(0.3) + \pi_{01}(0.24) + \pi_{10}(0.24) + \pi_{11}(0.2)$
$\pi_{10} = \pi_{00}(0.3) + \pi_{01}(0.24) + \pi_{10}(0.24) + \pi_{11}(0.2)$
$\pi_{11} = \pi_{00}(0) + \pi_{01}(0.36) + \pi_{10}(0.36) + \pi_{11}(0.6)$
$\pi_{00} + \pi_{01} + \pi_{10} + \pi_{11} = 1$
Solving above equations, we get,
$\pi_{00} = 0.1237, \pi_{10} = 0.2307, \pi_{01} = 0.2307, \pi_{11} = 0.41438$
$throughput = \pi_{00}[1] + (\pi_{01} + \pi_{10}) * 2 + \pi_{11}[1] = 0.1237 + (0.2307 + 0.2307) * 2 + 0.41438$
$= 1.4613\ pps$
$Throughput\ for\ output\ port\ 0 = \pi_{00}[1] + (\pi_{01} + \pi_{10}) * 2 + \pi_{11}[1]$
$= 0.1237 + 0.2307 * 2 = 0.5851$
$Throughput\ for\ output\ port\ 1 = \pi_{00}[1] + (\pi_{01} + \pi_{10}) * 2 + \pi_{11}[1]$
$= 0.2307 * 2 + 0.41438 = 0.87578$

$$Overall\ switch\ performance = \frac{1.46}{2} * 100 = 73.065\%$$

For $\alpha_0 = \alpha_1 = 0.5$

$$p = \begin{bmatrix} \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & 0 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \end{bmatrix}$$

$\pi_{00} = \pi_{00}\left(\frac{1}{2}\right) + \pi_{01}\left(\frac{1}{4}\right) + \pi_{10}\left(\frac{1}{4}\right) + \pi_{11}(0)$
$\pi_{01} = \pi_{00}\left(\frac{1}{4}\right) + \pi_{01}\left(\frac{1}{4}\right) + \pi_{10}\left(\frac{1}{4}\right) + \pi_{11}\left(\frac{1}{4}\right)$
$\pi_{10} = \pi_{00}\left(\frac{1}{4}\right) + \pi_{01}\left(\frac{1}{4}\right) + \pi_{10}\left(\frac{1}{4}\right) + \pi_{11}\left(\frac{1}{4}\right)$
$\pi_{11} = \pi_{00}(0) + \pi_{01}\left(\frac{1}{4}\right) + \pi_{10}\left(\frac{1}{4}\right) + \pi_{11}\left(\frac{1}{2}\right)$
$\pi_{00} + \pi_{01} + \pi_{10} + \pi_{11} = 1$

Solving above equations, we get,
$\pi_{00} = \pi_{10} = \pi_{01} = \pi_{11} = \frac{1}{4}$

$$throughput = \pi_{00}[1] + (\pi_{01} + \pi_{10}) * 2 + \pi_{11}[1] = \frac{1}{4} + \left(\frac{1}{4} + \frac{1}{4}\right) * 2 + \frac{1}{4} = 1.5\ pps$$

$$Throughput\ for\ output\ port\ 0\ = \pi_{00}[1] + (\pi_{01} + \pi_{10}) * 2 + \pi_{11}[1] = 0.25 + 0.25 * 2 = 0.75$$

$$Throughput\ for\ output\ port\ 1\ = \pi_{00}[1] + (\pi_{01} + \pi_{10}) * 2 + \pi_{11}[1] = 0.25 + 0.25 * 2 = 0.75$$

$$Overall\ switch\ performance = \frac{1.5}{2} * 100 = 75\%$$

Experimental results

For N=2 ($\alpha_0 = \{0.1, 0.2, 0.3, 0.4, 0.5\}$)

Simulation result for 2*2 Switch

```
start for 2 * 2 switch

PI values:  [0.00190019 0.05550555 0.05480548 0.88778878]
summation of pi values 1.0
0.05 0.47 0.47 0.00
0.02 0.09 0.08 0.81
0.01 0.09 0.08 0.82
0.00 0.05 0.05 0.90
Throughput of the switch for N = 2 is : 1.1103110311031101

PI values:  [0.0128071  0.11481881 0.11611854 0.75625555]
summation of pi values 1.0
0.21 0.43 0.36 0.00
0.05 0.16 0.15 0.65
0.04 0.14 0.15 0.67
0.00 0.10 0.10 0.80
Throughput of the switch for N = 2 is : 1.2309373538943298

PI values:  [0.04420442 0.18091809 0.1820182  0.59285929]
summation of pi values 1.0
0.30 0.36 0.34 0.00
0.08 0.21 0.21 0.49
0.09 0.20 0.21 0.50
0.00 0.15 0.15 0.69
Throughput of the switch for N = 2 is : 1.362936293629363

PI values:  [0.12378101 0.23171518 0.22831402 0.41618979]
summation of pi values 1.0
0.41 0.29 0.31 0.00
0.16 0.24 0.24 0.36
0.16 0.25 0.23 0.36
0.00 0.20 0.20 0.61
Throughput of the switch for N = 2 is : 1.4600291992869638

PI values:  [0.24192419 0.25262526 0.25222522 0.25322532]
summation of pi values 0.9999999999999999
0.49 0.25 0.26 0.00
0.25 0.26 0.24 0.25
0.24 0.24 0.26 0.25
0.00 0.25 0.25 0.50
Throughput of the switch for N = 2 is : 1.5048504850485052
```

We observe that the $\pi_{00} = \pi_{10} = \pi_{01} = \pi_{11}$ values obtained in the experiment are approximately equal to the values obtained theoretically. For 2*2 Balanced switch $i.e., \alpha_0 = 0.5$, Throughput of the system is equal to 1.504 which is equal to the theoretically value.

For N=4,

Simulation results for 4*4 switch

```
start for 4 * 4 switch

PI values:  [0.01032826 0.00500801 0.00590905 0.00440891 0.00580484 0.00290089
 0.00460185 0.00440233 0.00520533 0.00289734 0.00520155 0.0041011
 0.00530129 0.00369805 0.00310061 0.00320074 0.00601314 0.00559879
 0.00500162 0.00260146 0.00400166 0.00539965 0.00409976 0.00309993
 0.00460176 0.00399956 0.00330011 0.00429987 0.00380094 0.00289996
 0.00329953 0.0035997  0.00330453 0.00350132 0.00330145 0.00340215
 0.00360124 0.00410012 0.00330007 0.00219996 0.00460125 0.00319995
 0.00639975 0.00329987 0.00400171 0.0025997  0.00309995 0.0036
 0.00460688 0.00360093 0.00410242 0.00329994 0.00350086 0.00289985
 0.00219993 0.00179974 0.00350108 0.00230001 0.00269967 0.00390003
 0.00330045 0.00279989 0.00449987 0.00629901 0.00611383 0.00389863
 0.00369873 0.00329779 0.0049079  0.00449993 0.00380006 0.00320047
 0.00230116 0.00359696 0.00289986 0.00229955 0.00450703 0.00359973
 0.00350055 0.00330004 0.004204   0.00499946 0.00280047 0.00360013
 0.00390046 0.00749915 0.00469948 0.0050994  0.00440035 0.00499912
 0.00279972 0.00299952 0.00339998 0.00529906 0.00449935 0.00489947
 0.00420505 0.0030999  0.00320034 0.00309697 0.00330056 0.00499951
 0.00339973 0.00359968 0.00280045 0.00309954 0.00549935 0.00209971
 0.00440101 0.00309955 0.0020998  0.0042993  0.00260554 0.00429713
 0.00289696 0.00359978 0.00330036 0.00489948 0.00359975 0.0036995
 0.00330026 0.00329934 0.00409945 0.00329905 0.00410061 0.00409942
 0.00289962 0.00569886 0.00500827 0.00220048 0.00330274 0.00460054
 0.00379997 0.00419964 0.00329993 0.00269727 0.00350212 0.00279974
 0.00499996 0.00369393 0.00400082 0.00259706 0.00409996 0.0026001
 0.00330366 0.00379697 0.00290004 0.00329666 0.00370031 0.00499941
 0.00449909 0.00409945 0.00219982 0.00479936 0.00539953 0.00379963
 0.00329989 0.00319938 0.0030994  0.00429858 0.00359984 0.00290033
 0.00519661 0.00309974 0.00359978 0.00339934 0.00459918 0.0030994
 0.00339995 0.00449921 0.00869875 0.00489955 0.00349998 0.00379962
 0.00489921 0.00399932 0.00360079 0.00340064 0.00229647 0.00379609
 0.00259979 0.00289949 0.00329948 0.00279956 0.00270001 0.00399923
 0.00539946 0.00399929 0.0038995  0.00339939 0.00399961 0.00529872
 0.00541515 0.00350291 0.00400577 0.00340191 0.00420867 0.00420029
 0.00299961 0.00380004 0.00420073 0.00279973 0.00429994 0.00369969
 0.00360298 0.00279957 0.00360002 0.00519929 0.00380273 0.00419704
 0.00279709 0.00339981 0.00430018 0.00539925 0.00349973 0.0034993
 0.00299958 0.00259952 0.00349978 0.00359949 0.00320003 0.00399933
 0.00419922 0.00589904 0.00310044 0.00209393 0.00370006 0.0038998
 0.00260018 0.0038996  0.00249967 0.00379943 0.00320046 0.00379972
 0.00439958 0.00359944 0.0037003  0.00419923 0.00419949 0.00509923
 0.00340143 0.00319646 0.00339974 0.00549638 0.00339964 0.00469946
 0.00379927 0.00569879 0.00359973 0.00379956 0.00429923 0.00579805
 0.0061993  0.00569888 0.0062983  0.00809867]
summation of pi values 1.0000000000000002
Throughput of the switch for N = 4 is : 2.611539508009928
```

For N=4,

$throughput = 2.605$

$Overall\ switch\ performance = \frac{2.605}{4} * 100 = 65.05\%$

We observe that for N=4, the transition state matrix is computed. Throughput for N=4 comes out to be 2.605, and efficiency decreases to 65.05%.

Similarly, we simulate the results for N=8.

For N=8,

$throughput = 4.947$

$Overall\ switch\ performance = \frac{4.94}{4} * 100 = 61.74\%$

We observe that as N increases the Overall switch performance for $N * N\ switch$ goes on decreasing.

Hotspot Traffic

For N=4,

$\alpha_1 = \frac{1}{k}, \alpha_j = \left(\frac{1}{N-1}\right)\left(\frac{k-1}{k}\right)\ for\ j \neq 1$

States;

[3 2 2 2]
[3 3 1 2]
[0 2 2 2]
[3 1 0 2]
[0 2 0 1]
[0 2 1 2]
[2 2 0 2]
[0 1 1 3]
[1 0 3 3]
[1 0 2 2]
[0 1 2 1]
[2 0 1 1]
[3 1 2 1]
[1 2 3 2]
[3 1 0 0]
[1 0 2 3]
[1 2 0 3]
[1 3 2 1]
[3 1 2 3]
[2 0 2 2]
[1 2 2 2]
....
.....

iv. Reference

- http://www-personal.umich.edu/~gaozheng/teaching/stats414/Simulation/simulation.html
- http://www.columbia.edu/~ks20/stochastic-I/stochastic-I-MCII.pdf
- https://stats.stackexchange.com/questions/328940/monte-carlo-method-and-convergence-in-distribution

v. Source Code

```
 import random
import collections
import numpy as np
import itertools
from discreteMarkovChain import markovChain

def t_matrix(trans, k):
    n = k                   #number of states
    M = [[0]*n for _ in range(n)]
    for (i,j) in zip(trans,trans[1:]):
        M[i][j] += 1

    #convert to probabilities:
    for row in M:
        s = sum(row)
        if s > 0:
            row[:] = [float(f)/float(s) for f in row]
    return M

def pRange(n):
    alphaL = [0] * (n+1)
    alphai = 1.0/n
    incAlpha = 0
    for a in range(n+1):
        alphaL[a] = incAlpha
        incAlpha = incAlpha + alphai
    return alphaL

def repeated_op(inp):
    return [item for item, count in collections.Counter(inp).items() if count > 1]

def Head_of_line(n, inpList):
    #input list to the switch
    #print "Input list before the cycle: ", inpList
    oList = [0] * n
    #differents outputs
   # rang = range(1, n+1, 1)
    #send the values to the outputs which have only one input --
    # which wants repeated outputs --
    repeatoutput = repeated_op(inpList)
```

```
    #Empty singly mapped i/o o/p ie they have been sent to O/P --
    for i,j in enumerate(inpList):
        if j not in repeatoutput:
            oList[j-1] = i+1
            inpList[i] = 0
    #Use RNG to decide which repeated I/P to transition to O/P
    listRepeat = {}
    for inp in inpList:
        if inp != 0 and inp not in listRepeat:
            listRepeat[inp] = [k for k, y in enumerate(inpList) if y == inp]
    #print "repeated numbers location in the list : %s "% listRepeat
    for b in listRepeat:
        p = pRange(len(listRepeat[b]))
        #print "probability division for repeated number %s, = %s"%(b ,p)
        #Gen rand number to decide which to select
        ran = random.random()
        #compare the ran with p and listRepeat and remove that value from input list
        #print "input list and output list state", inpList, oList
        prop = np.digitize(ran, p)
        #print "number to be propd, ran", prop, ran
        # np is the location of the number in the input list which needs to be delivered to the output
        prop = listRepeat[b][prop-1]
        oList[inpList[prop]-1] = prop +1
        inpList[prop] = 0
    #print "input and output values after a cycle: ",inpList, oList
    return inpList, oList

inp = [2, 4]
traffic = [2,"balanced"]
#traffic = [2, "balanced", "unbalanced"]
for i in inp:
    inMatrix = []
    states = i**i
    print("start for %s * %s switch \n"%(i,i))
    # create a n x n matrix transition matrix
    #tMatrix = [[0 for x in range(states)] for y in range(states)]

    rang = range(1, i+1, 1)
    inputList_temp = []
    inMatrixTemp = []
    rangeList = []
    for inp in rang:
        #list inpList = list(i) wants to go to the particular output
        #initial input value
        inputList_temp.append(random.choice(rang))
    #print "input List = ",inpList
    #values after each cycle
    for traff in traffic:
        inMatrix = []
        #for n = 2 and alpha = [0, 0.1,0.2,0.3,0.4,0.5]
```

```
if traff == 2 and inp == 2:
    #alpha1 = [0]
    alpha1 = [0.1, 0.2, 0.3, 0.4, 0.5]
    for al in alpha1:
        inMatrixTemp = []
        for a in range(10000):
            inMatrixTemp.append(inputList_temp[:])
            inList, outList = Head_of_line(i, inputList_temp[:])
            #get the number of 0's and their location
            for z, nul in enumerate(inList):
                if nul == 0:
                    ranChoice = random.random()
                    if ranChoice < al:
                        inList[z] = 1
                    else:
                        inList[z] = 2
                    #choose according to different values of alpha
            inputList_temp = inList
        inMatrix.append(inMatrixTemp[:])

 #For Balanced Traffic alpha = 1/n
 elif traff == "balanced" and inp != 2:
    inMatrixTemp = []
    for a in range(10000):
        inMatrixTemp.append(inputList_temp)
        inList, outList = Head_of_line(i, inputList_temp[:])
        for z, nul in enumerate(inList):
            if nul == 0:
                inList[z] = random.choice(rang)
        inputList_temp = inList
    inMatrix.append(inMatrixTemp)

#     for i in range(0,25):
 #           print(np.random.randint(low=0,high=4,size=4))
  elif traff == "unbalanced" and inp != 2: #unbalanced traffic alpha1 = 1/k alphaj = (1/n-1)(k-1/k)
    kList = range(2, i-1, 1)
    #kList = [4]
    for kk in kList:
        inMatrixTemp = []
        alphaL = [0]
        alphaL.append(1.0/kk)
        qVal = (1.0/(i-1))*((kk-1)/float(kk))
        #alphaL.remove(0)
        for g in range(i):
            alphaL.append(alphaL[g+1]+qVal)
       #  print "aaaaaaaaaa",alphaL
        for a in range(10000):
            inMatrixTemp.append(inputList_temp[:])
            inList, outList = Head_of_line(i, inputList_temp[:])
            for z, nul in enumerate(inList):
```

```
                if nul == 0:
                    rand = np.random.random_sample((1))
                    inList[z+1] = int(np.digitize(rand, alphaL))
                inputList_temp = inList
            for i in range(0,25):
                print(np.random.randint(low=0,high=4,size=4))
            inMatrix.append(inMatrixTemp[:])
        #for a in inList:
        #    if a == 0:
    for inMatrix in inMatrix:
        #print inMatrix
        RangeOfInputs = range(1, i+1)
        cartesianProduct = list(itertools.product(RangeOfInputs, repeat = i))
        #print cartesianProduct
        efficiancyMatrix = [] #has info about what combination has what OP
        stateThroughputCount = 0
        #calculate throughput multiplier for each state using the cartesian product
        for throu in cartesianProduct:
            stateThroughputCount = 0
            rep = repeated_op(list(throu))
            for thr in list(throu):
                if thr not in rep:
                    stateThroughputCount = stateThroughputCount + 1
            stateThroughputCount = stateThroughputCount + len(rep)
            efficiancyMatrix.append(stateThroughputCount)
            #print stateThroughputCount, throu
       # print efficiencyMatrix, cartesianProduct
        relationShipList = []
        #map from cartesian product to the input list and store as a list of transitions
        for inMat in inMatrix:
            for d, lis in enumerate(cartesianProduct):
                if lis == tuple(inMat):
                    relationShipList.append(d)

            #i, lis = [(i, lis) for i, lis in enumerate(cartesianProduct) if lis == inMat]
            #print i, lis
        tMatrix = t_matrix(relationShipList, len(cartesianProduct))
        numPYArray = np.array(tMatrix)
        mc = markovChain(numPYArray)
        mc.computePi('linear')
        print ("PI values: ", mc.pi    )
        print("summation of pi values", sum(mc.pi))
        if i == 2:
            for row in tMatrix: print(' '.join('{0:.2f}'.format(x) for x in row))
        #print "relationShipList", relationShipList

        #to calculate the overall efficiancy of the switch, multiply the pi values with its corresponding efficiancyMatrix
values
        Throughput = sum([a*b for a,b in zip(mc.pi,efficiancyMatrix)])
        print("Throughput of the switch for N = %s is : %s \n"% (i, Throughput) )
```