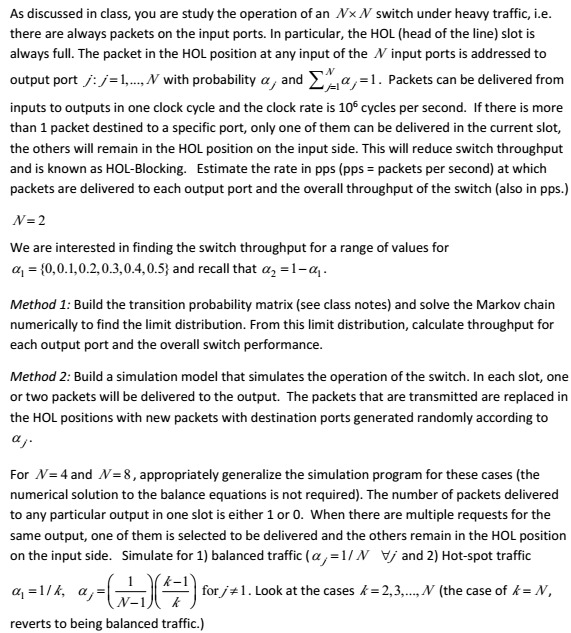
**Project 4：Switch Performance and HOL Blocking**

**EE 511 – Section** Tuesday 5:00pm—5:50pm

**Name: Yiqi Feng**

**Student ID #:** 9057129035

***1. Problem Statement***



***2. Theoretical Analysis***

For N=2 (Method1), assume π(t) = [ (t),(t), (t),(t)], where (t) means the probability that the packet from the input port 0 is addressed to output port i and the packet from the input port 1 is addressed to output port j.

The transition probability matrix P is:

p (1−p)/3 (1−p)/2 0

p2 p(1−p) p(1−p) (1−p)2

p2 p(1−p) p(1−p) (1−p)

1. p/2 p/2 1−p

And we have the transition equation:

π(t + 1) = π(t)P

By solving the equation set, we can get the limiting distribution:

π = πP = 1

After we get the limiting probability, we can easily get the throughput and the efﬁciency: throughput = (×1 + ×2 + ×2 + ×1)×cycle/second

efficiency = throughput 2×cycle/second

For N=2 (Method2), we have two main variables: s and pp, which are both vectors with two elements. s refers to the output port that the packet will be sent to. For example, s[0] = 1 means the packet from input port 0 will be sent to the output 1; s[1] = 1 means the packet from input 1 will be sent to the output port 1. pp refers to the accumulative number of packets which are sent to a certain output port. For example, pp[0] = 1000 means 1000 packets have already gone out through output port 0; pp[1] = 2000 means 2000 packets have already gone out through output port 1. In each cycle, the s and pp variable will be updated. For example, suppose at time t, we have:

s[0] = 1,s[1] = 1,pp[0] = 1000,pp[1] = 2000

During this cycle, two packets have the same output port and only one packet can be sent to that port. We assume the winner is the packet from input port 1. Then pp[1] is increased by 1 and pp[0] remains its value. At the same time, the input port 1 will have a new packet and its destination is random: 0 with probability p or 1 with probability 1−p. Again, we assume the new destination is 0. Finally, s and pp have their new value:

s[0] = 1,s[1] = 0,pp[0] = 1000,pp[1] = 2001

For N, we still have two main variables: s and pp, and they are vectors with N(N ≥ 2) elements. However, besides s and pp, we have another important variable M, which is a dictionary data structure with key and value. The meaning of s and pp has no changes. Let’s talk about M. M stores the input ports whose packets will be sent to a certain output port. The key of M is the output port number, and the value of M is a list of input port number. For example, we have M[2] = [1,3,4,5], which means the packets from input port 1, 3, 4, and 5 have the same destination, which is output port 2. In every cycle, we need to update three variables s, pp and M. Suppose N = 3, at a time point, we have following variables:

s[0] = 1, s[1] = 1, s[2] = 2

According to the s, the M variable is:

M[0] = [], M[1] = [0,1], M[2] = [2]

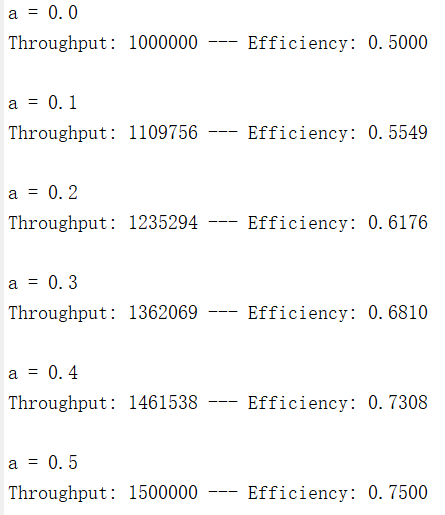
We suppose the packet from input port 0 is chosen to send to output port 1 and the new packet arriving at input port 0 is destined to output port 0. At the same time, the packet from input port 2 is sent to output port 2, which is a deterministic event. The new arriving packet at input port 2 is random and we suppose its destination is output 1. Now the M is changed:

M[0] = [0], M[1] = [0,2], M[2] = []

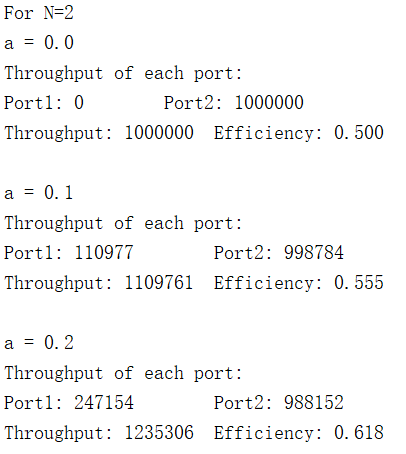
In each cycle, some ports may be added to the port list of M[i] and some ports may be removed from the port list of M[i]. (i = 1,2...,N)

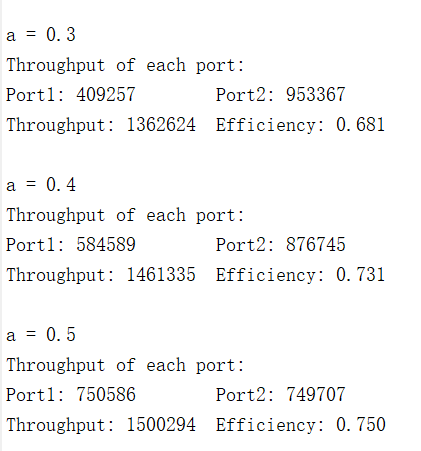
***3. Experiments and Results***

For N=2(method 1):

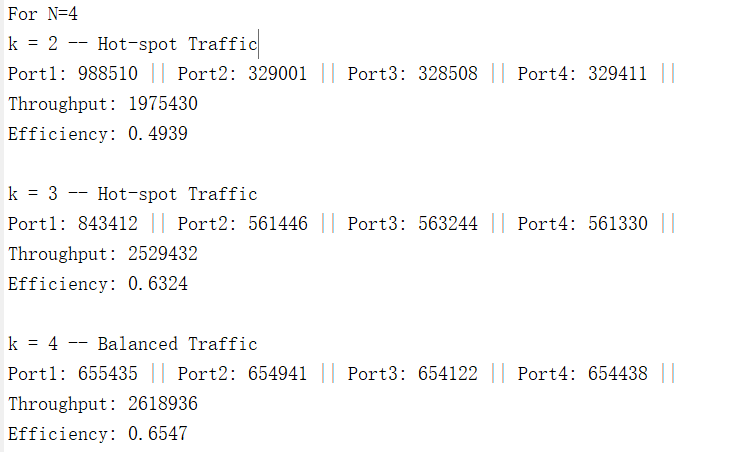


For N=2(method 2):

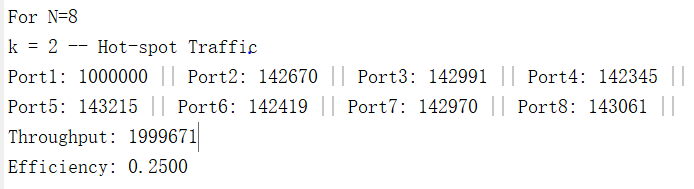


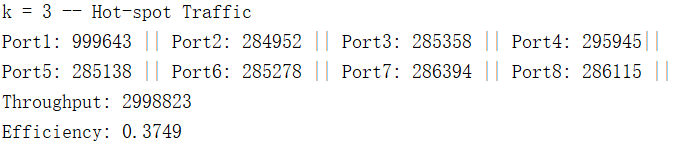


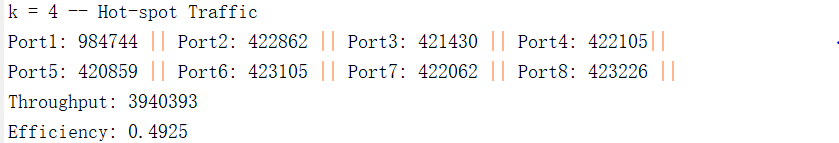
For N=4:

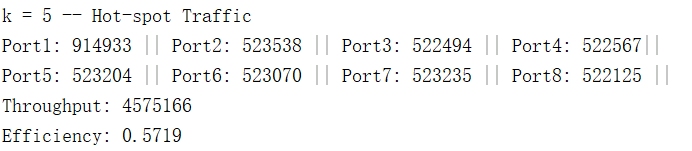


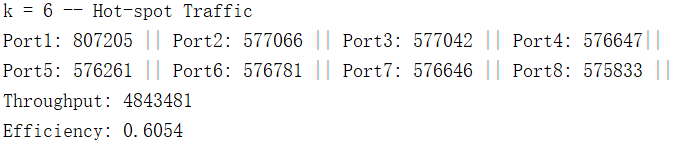
For N=8:

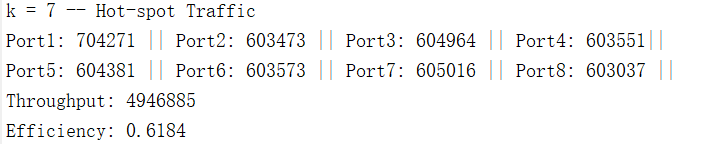




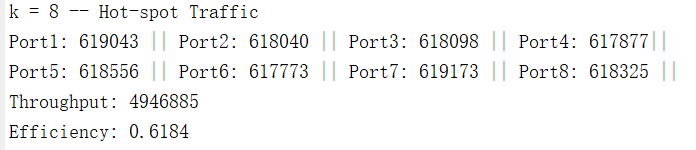












***4. Source Code***

The code for N=2(method 1):

a = [0, 0.1, 0.2, 0.3, 0.4, 0.5];

oneSecond = 10^6;

res = zeros(length(a), 4);

for i = 1 : length(a)

p = a(i);

matrix = [p, (1-p)/2, (1-p)/2, 0;

p^2, p\*(1-p), p\*(1-p), (1-p)^2;

p^2, p\*(1-p), p\*(1-p), (1-p)^2;

0, p/2, p/2, 1-p];

syms x1 x2 x3 x4

sol = [x1, x2, x3, x4];

eqns = [sol == sol \* matrix, sum(sol) == 1];

re = solve(eqns, sol);

res(i, 1) = re.x1;

res(i, 2) = re.x2;

res(i, 3) = re.x3;

res(i, 4) = re.x4;

throughput = oneSecond \* (1 \* res(i, 1) + 2 \* res(i, 2) + 2 \* res(i, 3) + 1 \* res(i, 4));

efficiency = throughput / (2 \* oneSecond);

fprintf('a = %.1f\n', p);

fprintf('Throughput: %d --- Efficiency: %.4f\n', round(throughput), efficiency);

fprintf('\n');

end

The code for N=2(method 2):

a = [0, 0.1, 0.2, 0.3, 0.4, 0.5];

oneSecond = 10^6;

numberOfSecond = 5;

fprintf('For N=2');

fprintf('\n');

for i = 1 : length(a)

p = a(i);

s = [1, 2];

pp = zeros(1, 2);

second = 1;

cycle = 1;

while second <= numberOfSecond

while cycle <= oneSecond

if s(1) ~= s(2)

pp(1) = pp(1) + 1;

pp(2) = pp(2) + 1;

s(1) = nextPacket(2, p);

s(2) = nextPacket(2, p);

else

pp(s(1)) = pp(s(1)) + 1;

winner = randi(2);

if winner == 1

s(1) = nextPacket(2, p);

else

s(2) = nextPacket(2, p);

end

end

cycle = cycle + 1;

end

second = second + 1;

cycle = 1;

end

pp = pp ./ numberOfSecond;

fprintf('a = %.1f\n', p);

fprintf('Throughput of each port: \n')

for j = 1 : 2

fprintf('Port%d: %d ', j, round(pp(j)));

end

fprintf('\n');

throughput = sum(pp);

fprintf('Throughput: %d ', round(throughput));

efficiency = throughput / (2 \* oneSecond);

fprintf('Efficiency: %.3f\n', efficiency);

fprintf('\n');

end

The code for N=4 or N=8:

N = 4; % N could be changed

oneSecond = 10^6;

numberOfSecond = 1;

fprintf('For N=%d',N);

for i = 2 : N

p = 1/i;

s = zeros(1, N);

for j = 1 : N

s(j) = j;

end

pp = zeros(1, N);

second = 1;

cycle = 1;

key = zeros(1, N);

for j = 1 : N

key(j) = j;

end

val = cell(1, N);

for j = 1 : N

val{j} = [0, j];

end

M = containers.Map(key, val);

while second <= numberOfSecond

while cycle <= oneSecond

add = zeros(N, 2);

addNum = 1;

for j = 1 : N

candidate = M(j);

total = length(candidate);

if total == 1

continue;

end

winner = randi(total-1) + 1;

newPacket = nextPacket(N, p);

s(candidate(winner)) = newPacket;

add(addNum, 1) = newPacket;

add(addNum, 2) = candidate(winner);

addNum = addNum + 1;

candidate(winner) = [];

M(j) = candidate;

pp(j) = pp(j) + 1;

end

for j = 1 : addNum - 1

port = add(j, 1);

new = add(j, 2);

M(port) = [M(port), new];

end

cycle = cycle + 1;

end

second = second + 1;

cycle = 1;

end

pp = pp ./ numberOfSecond;

fprintf('k = %d -- ', i);

if i ~= N

fprintf('Hot-spot Traffic\n');

else

fprintf('Balanced Traffic\n')

end

fprintf('Throughput of each port: \n')

for j = 1 : N

if mod(j, 5) == 0

fprintf('\n');

end

fprintf('Port%d: %d || ', j, round(pp(j)));

end

fprintf('\n');

throughput = sum(pp);

fprintf('Throughput: %d\n', round(throughput));

efficiency = throughput / (N \* oneSecond);

fprintf('Efficiency: %.4f\n', efficiency);

fprintf('\n');

end