For a 2x2 switch,

A picture containing sky, boat

Description generated with high confidence

A picture containing boat, sky, map

Description generated with very high confidence

Therefore, when p > 0.7, the efﬁciency seems to be steady to be around 0.75. We can conclude that if we choose the p to be 0.7, the packet may not pile up in the buffer and still the switch has a good efﬁciency. To compute the 95% conﬁdence interval we choose p from 0.1 to 0.9,

|  |  |
| --- | --- |
| p | CI efﬁciency (%) |
| 0.1 | [6.25,12.5] |
| 0.2 | [15.50,23.25] |
| 0.3 | [25.50,33.75] |
| 0.4 | [35.50,45.50] |
| 0.5 | [44.50,54.25] |
| 0.6 | [54.25,64.25] |
| 0.7 | [65.75,72.50] |
| 0.8 | [70.0,77.0] |
| 0.9 | [71.0,78.0] |

A picture containing sky

Description generated with high confidence

We just change the parameter r1,r2 in the function and we can get the switch’s behaviour when r1 = 0.75 and r2 = 0.25.

Clearly, the efﬁciency decreases and it is more easily for the buffer to pile up packets.A close up of a map

Description generated with high confidence

|  |  |
| --- | --- |
| p | CI efﬁciency (%) |
| 0.1 | [7.75,12.75] |
| 0.2 | [16.75,23.75] |
| 0.3 | [25.50,34.50] |
| 0.4 | [35.25,45.25] |
| 0.5 | [43.75,53.75] |
| 0.6 | [55.25,62.75] |
| 0.7 | [60.75,67.00] |
| 0.8 | [61.25,65.50] |
| 0.9 | [60.75,70.55] |

Mean\_N1\_pkt = 0.1573

Mean\_Throughput = 0.8754

A screenshot of a cell phone

Description generated with high confidence

A close up of a map

Description generated with high confidence

The distribution of the number of packets in buffer is a geometric distribution; and to my expectation, the distribution of the number of packets processed per slot is a normal distribution. This can be explained by the Central limit theorem.

For a 4x4 switch with balanced traffic, throughput = 2.6060 pps

For a 8x8 switch with balanced traffic, throughput = 4.9820 pps

5. CODES

For 2x2 Switch

(i)

function [nstate, efficiency1, efficiency2] = Input(p,r1,r2,tend)

%

% nstate: record how many packets are present in input1 and input2

% p: the arriving probability

% r1: the probability to target at output 1

% r2: the probability to target at output 2

% tend: continue until tend

% nstate how many packet in each of the input

nstate = [p > rand(1) , p > rand(1) ];

passed = 0;

maxpassed = 0;

% r1 = 0.5; r2 = 0.5;

% nInput1 = nstate(1);

attempt = [r1 > rand(1) , r1> rand(1)];

nextAttempt = [r1 > rand(1) , r1> rand(1)];

% nInput2 = nstate(2);

priority = 1;

t = 1;

while t<tend

if sum(nstate) > 0

if nstate(1) > 0 && nstate(2) == 0

nstate(1) = nstate(1) - 1;

passed = passed + 1;

maxpassed = maxpassed + 1;

else

if nstate(1) == 0 && nstate(2) > 0

nstate(2) = nstate(2) - 1;

passed = passed + 1;

maxpassed = maxpassed + 1;

else % when both have packets at the input

if attempt (1) ~= attempt (2)

nstate(1) = nstate(1) - 1;

nextAttempt = [r1 > rand(1) , r1 > rand(1)];

passed = passed + 2;

maxpassed = maxpassed + 2;

else

if priority == 1

nstate(1) = nstate(1) - 1;

priority = - priority;

nextAttempt = [r1 > rand(1) , attempt(2)];

else

nstate(2) = nstate(2) - 1;

priority = - priority;

nextAttempt = [attempt(1) , r1 > rand(1)];

end

passed = passed + 1;

maxpassed = maxpassed + 2;

end

end

end

% else

% nextAttempt = [r1 > rand(1) , r2 > rand(1)];

end

% New packet arrives

nstate = nstate + [p>rand(1) ,p>rand(1)];

attempt = nextAttempt;

t = t+1;

% efficiency = passed/t;

% nstate

end

efficiency1 = passed/maxpassed;

efficiency2 = passed/(t\*2);

(ii)

buffer = zeros(1,100);

k = 1;

for p = 0.1:0.05:1

for i = 1:100

[nstate,efficiency1,efficiency2] = Input(p,0.5,0.5,200);

buffer(i) = sum(nstate);

efficiency(i) = efficiency2;

end

recordNumber(k) = mean(buffer);

recordEfficiency(k) = mean(efficiency);

k = k+1;

end

figure,

plot (0.1:0.05:1 ,recordNumber);

title ( 'Packets in buffer given different p (200 time slots)');

xlabel ( 'p' );

ylabel( 'Number of packet in buffer after 200 time slots');

figure,

plot (0.1:0.05:1 ,recordEfficiency);

title ( 'Efficiency given different p(200 time slots)');

xlabel ( 'p' );

ylabel( 'Efficiency (\times 2 is the number of packet processed per time slot)');

(iii)

close all;

clear;

clc;

times = 1000;

casei = 1;

for p = 0:0.1:1

N1\_pkt = zeros(1,times);

N2\_pkt = zeros(1,times);

throughput = zeros(1,times);

for i = 1:times

[N1\_pkt(i), N2\_pkt(i), throughput(i)] = Load\_Data(p,casei);

end

p

Mean\_N1\_pkt = mean(N1\_pkt)

Mean\_throughput = mean(throughput)

end

(iv)

function [N1,N2,throughput] = Load\_Data(p, casei)

n\_slot = 1000;

P1 = rand(1,n\_slot);

P2 = rand(1,n\_slot);

R11 = rand(1,n\_slot);

R21 = rand(1,n\_slot);

% Packets processed per slot

pps = zeros(1,n\_slot);

throughput = 0;

N1 = 0; % Number of packets in buffer 1

N2 = 0; % Number of packets in buffer 2

if casei == 1

r = 0.5;

else

r = 0.75;

end

for i = 1:n\_slot

% Situation that input 1 and input 2 both have packets

if(P1(i)<p && P2(i)<p)

% Input 1 switches to output 1 and input 2 switches to output 2

if(R11(i)<r && R21(i)>r)

pps(i) = 2;

[N1] = popbuffer(N1);

[N2] = popbuffer(N2);

end

% Input 1 switches to output 1 and input 2 switches to output 2

if(R11(i)>r && R21(i)>r)

P\_sel = rand(1);

% Select packet with the same probability

if P\_sel <= 0.5

pps(i) = 1;

[N1] = popbuffer(N1);

[N2] = pushbuffer(N2);

else

pps(i) = 1;

[N1] = pushbuffer(N1);

[N2] = popbuffer(N2);

end

end

% Input 1 switches to output 2 and input 2 switches to output 1

if(R11(i)>r && R21(i)<r)

pps(i) = 2;

[N1] = popbuffer(N1);

[N2] = popbuffer(N2);

end

% Input 1 switches to output 1 and input 2 switches to output 1

if(R11(i)<r && R21(i)<r)

P\_sel = rand(1);

% Select packet with the same probability

if P\_sel <= 0.5

pps(i) = 1;

[N1] = popbuffer(N1);

[N2] = pushbuffer(N2);

else

pps(i) = 1;

[N1] = pushbuffer(N1);

[N2] = popbuffer(N2);

end

end

end

% Situation that input 1 has packet but input 2 doesn't

if(P1(i)<p && P2(i)>p)

pps(i) = 1;

[N1] = popbuffer(N1);

end

% Situation that input 2 has packet but input 1 doesn't

if(P1(i)>p && P2(i)<p)

pps(i) = 1;

[N2] = popbuffer(N2);

end

% Situation that input 1 and input 2 both don't have packet

if(P1(i)>p && P2(i)>p)

pps(i) = 0;

end

end

throughput = sum(pps)/n\_slot;

end

(v)

function [N1] = popbuffer(N1)

% if the number of packets in buffer > 1 then pop one packet out

% if number <= 0 then number becomes 0 after pop

if N1>1

N1 = N1 - 1;

else

N1 = 0;

end

(vi)

function [N1] = pushbuffer(N1)

% Push one packet into buffer, number increases by 1

N1 = N1 + 1;

End

(vii)

close all;

clear;

clc;

times = 1000;

p = 0.5;

casei = 1;

N1\_pkt = zeros(1,times);

N2\_pkt = zeros(1,times);

throughput = zeros(1,times);

for i = 1:times

[N1\_pkt(i), N2\_pkt(i), throughput(i)] = Load\_Data(p,casei);

end

Mean\_N1\_pkt = mean(N1\_pkt)

figure,

A = 0:1:5;

[a,b] = hist(N1\_pkt,A);

bar(b,a/sum(a));

grid on;

title('Histogram of the number of packets in the buffer at input 1');

xlabel('Number of Packets');

ylabel('Frequency');

Mean\_throughput = mean(throughput);

figure,

A = unique(throughput);

dist\_throughput = histc(throughput,A);

a = dist\_throughput/sum(dist\_throughput);

plot(A,a);

grid on;

title('Distribution of the number of packets processed per slot by the switch');

xlabel('PPS');

ylabel('Frequency');

(viii)

close all;

clear;

clc;

times = 1000;

casei = 1;

for p = 0:0.1:1

N1\_pkt = zeros(1,times);

N2\_pkt = zeros(1,times);

throughput = zeros(1,times);

for i = 1:times

[N1\_pkt(i), N2\_pkt(i), throughput(i)] = Load\_Data(p,casei);

end

p

Mean\_N1\_pkt = mean(N1\_pkt)

Mean\_throughput = mean(throughput)

End

Generalizing to NxN switch

(i)

Balanced Traffic :

close all;

clear;

clc;

prompt = 'What is the size of the switch?';

N = input(prompt)

Hash = zeros(1,N);

total = 0;

steps = 1000;

num = N;

for j = 1:steps

empty = randi([1,N],1,num)

for i = 1:num

Hash(empty(i)) = Hash(empty(i)) + 1;

end

num = 0;

for i = 1:N

if Hash(i) > 0

Hash(i) = Hash(i) - 1;

num = num + 1;

end

end

total = total + num;

end

pps = total/steps;

(ii)

Hot-Spot Traffic :

close all;

clear;

clc;

prompt = 'What is the size of the switch?';

N = input(prompt)

Hash = zeros(1,N);

total = 0;

steps = 1000;

num = N;

k = 2;

for j = 1:steps

empty = randsrc(1,num,[1,2,3,4;1/k,(k-1)/(k\*(N-1)),(k-1)/(k\*(N-1)),(k-1)/(k\*(N-1))])

for i = 1:num

Hash(empty(i)) = Hash(empty(i)) + 1;

end

num = 0;

for i = 1:N

if Hash(i) > 0

Hash(i) = Hash(i) - 1;

num = num + 1;

end

end

total = total + num;

end

pps = total/steps;