Network Engineering - Mid-Term exam - Group 1 - 2021/2022

Preliminaries: x1 = 1st number of your NIA; x2 = 2nd number of your NIA; x3 = 3rd number of your NIA; x4 = 4th number of your NIA, etc. If any number is zero, consider it equal to 1.

Problem 1 (30 mins) - 5 points

Consider the Wi-Fi network depicted in Figure 1. The contention window is CW $= 8 \cdot x_2$. Other parameters include: $T_e = 0.9$ μs, and $L = 5000 \cdot x_3$ bits. Full-buffer traffic model is considered.

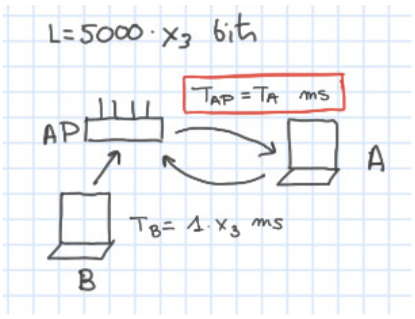

Figure 1: A Wi-Fi network that consists of 1 AP and 2 stations.

1. (1 point) Write your NIA, and calculate the different parameters that depend on it (transmission durations and CW). How many nodes are contending to access the channel?
2. (2 points) Considering that $T_A = T_{\text{AP}} = T_B = 1 \cdot x_3$ ms, calculate the transmission probability τ, the slot probabilities p_0, p_1, p_{1+}, and the throughput of the Access Point. Note that the throughput of the Access Point is the aggregate throughput of the Wi-Fi network divided by the number of contenders.
3. (2 points) Considering that $T_A = T_{\text{AP}} = 3.5 \cdot x_3$ ms (i.e., station A goes away from the AP), repeat previous point. Note that in this situation we can have successul transmissions of duration T_A and T_B, with probability $p_{1,A}$ and $p_{1,B}$. In case of collisions, note that in all cases they will have a duration of T_A.

Problem 2 (30 mins) - 5 points

We aim to study a Web server that consists of a single buffer of size $Q = \min(2, x_5)$, and two CPUs, A and B. CPU A is able to process up to μ_A requests/second, and CPU B is able to process up to $\mu_B < \mu_A$ requests/second. When the system is empty, an arriving request is always placed in CPU A. Moreover, when there are two requests in the system, and one departs, the other is moved to CPU A in case it was in CPU B with a negligible delay.

1. (1 points) Draw the Markov chain for the system described, write its local balance equations, and write its equilibrium distribution.
2. (2 points) Calculate the blocking probability of the Web server if $\lambda = 100 \cdot x_2$ request/minutes, $\mu_A = 80 \cdot x_3$ request/minute, and $\mu_B = 20 \cdot x_3$ requests/minute. Calculate also both the average waiting delay and the total delay suffered by incoming requests ($E[D_q]$ and $E[D]$, respectively) in the Web server.
3. (2 points) Compare previous system with the opposite one, i.e., when the system is empty, an arriving request is always placed in CPU B. Moreover, when there are two requests in the system, and one departs, the remaining one is moved to CPU B in case it was in CPU A.

```
function Exercise1()

% NIA
x1=1;
x2=2;
x3=3;
x4=4;

% Parameters
CW=8*x3;
Te=9E-6;
L=5000*x3;
T1_B = 1*x3*1E-3;

% 1)

N = 3; % Number of contenders
disp('Parameters and Contenders')
disp([T1_B CW N]);

% 2)

T1_A_AP = T1_B;

% Transmission probability
tau = 2/(CW+1); % ok also if tau = 2/(CW+2)
% Slots probability
p0 = (1-tau)^N;
p1 = N*tau*(1-tau)^(N-1);
pc = 1-p0-p1;

% Wi-Fi Throughput
T1_a = T1_B;
S_WiFi_a = L*p1/(p0*Te+(1-p0)*T1_a);
S_AP_a = S_WiFi_a/N;

disp('AP Throughput');
disp(S_AP_a);

% c)

T1_A_AP = 3.5*x3*1E-3;

p0 = (1-tau)^N;
% We split p1 in two
p1a = 2*tau*(1-tau)^(N); % succ tx of AP and A
p1b = tau*(1-tau)^N; % succ tx of B
pc = 1-p0-p1a-p1b;

S_WiFi_b = L*p1/(p0*Te+p1a*T1_A_AP+p1b*T1_B+pc*T1_A_AP);

S_AP_b = S_WiFi_b/N;

disp('Throughput: Same rate all | Different rates');
disp([S_AP_a S_AP_b]);

end


function Exercise2()

%NIA
```

```
x1=2;
x2=3;
x3=4;
x4=2;
x5=4;

% Buffer size
Q=min(2,x5); % it could 1 or 2...

% a) Markov chain (x5=4 --> Q=2)

% lambda, lambda, lambda, lambda, -
% (0) <-> (1) <-> (2) <-> (3) <-> (4)
% - , muA, muA+muB, muA+muB, muA+muB

% b)

% Values
lambda=100*x2;
muA=80*x3;
muB=20*x3;

% Stationary prob.
a=lambda/muA;
b=lambda/(muA+muB);

pi0 = 1 / (1 + a + b*a +b^2*a + b^3*a);
pi1 = a *pi0;
pi2 = a*b *pi0;
pi3 = a*b^2 *pi0;
pi4 = a*b^3 *pi0;

disp('Stationary distribution');
disp([pi0 pi1 pi2 pi3 pi4]);

% Average number of requests in the Web server, and buffer
EN = pi1*1 + pi2*2 + pi3*3 + pi4*4;
ENq = pi3*1 + pi4*2;
% Little to get the delay
ED = EN / (lambda*(1-pi4));
EDq = ENq / (lambda*(1-pi4));
disp('Delays: E[D] | E[Dq]');
disp([ED EDq]);

% Throughput?
disp('Blocking Probability');
disp(pi4);

% 3) The opposite system

a=lambda/muB;
b=lambda/(muA+muB);

pi0 = 1 / (1 + a + b*a +b^2*a + b^3*a);
pi1 = a *pi0;
pi2 = a*b *pi0;
pi3 = a*b^2 *pi0;
pi4 = a*b^3 *pi0;

disp('Stationary distribution');
disp([pi0 pi1 pi2 pi3 pi4]);

% Average number of requests in the Web server, and buffer
```

```
EN = pi1*1 + pi2*2 + pi3*3 + pi4*4;
ENq = pi3*1 + pi4*2;
% Little to get the delay
ED = EN / (lambda*(1-pi4));
EDq = ENq / (lambda*(1-pi4));
disp('Delays: E[D] | E[Dq]');
disp([ED EDq]);

% Throughput?
disp('Blocking Probability (opposite case)');
disp(pi4);

% The opposite system goes worse... more delay, and less throughput... why?
% Well, the system uses the faster CPU more often in the first case, which
% improves all other metrics.
% --> Better to do your best from the beginning than taking things easy...
% thinking there will be always time in the future to work hard.

end
```