Assignment #2
Impact of transmission errors in the performance of a wireless Network Link

Paulo Vasconcelos - 84987
Pedro Teixeira - 84715
University of Aveiro

February 26, 2020

Task 1

Task 1.a - probability of the link being in the interference state and in the normal state when one control frame is received with errors

Matlab Code

```
probs = [ 0.99; 0.999; 0.9999; 0.99999];
n_rate = 10^-7;
i_rate = 10^-3;
n_bits = 128 * 8;
n = n_bits;
i = 0;
p_n = 1 - (nchoosek(n, i) * n_rate ^i * (1-n_rate)^(n-i)); % probability of error in normal
p_i = 1 - (nchoosek(n, i) * i_rate ^i * (1-i_rate)^(n-i)); % prob. of error in interference
p_e = p_n * probs + p_i * (1-probs);

p_ns = p_n*probs ./ p_e;                        % probability of link in normal state
p_is = p_i*(1-probs) ./ p_e;                    % probability of link in interference state
```

Code Analysis

We created a probabilities vector (probs) used throughout the first task exercises. We also stored the bit error rate value of both states and the number of bits in a package as established by the exercise.
After dealing with the constant values, we calculated the probability of error in, at least, one bit in a package in both states. This was done by calculating the probability of a package arriving without any errors and obtaining its complementary value (1-x).
We used those values to calculate the probability of a package having, at least, one bit wrong. This was done using the calculated probabilities and the probability of each state.
Lastly, we calculated the probability of the link being in either state when a control frame is received with errors. This was possible through Bayes' Law.

Results

p	p(normal)	p(interference)
p = 99%	0.0156	0.9844
p = 99.9%	0.1376	0.8624
p = 99.99%	0.6150	0.3850
p = 99.999%	0.9411	0.0589

Conclusions

With the increase of the probability of the link being in normal state, there's also a significant increase of the probability of the errors being registered in normal state. This comes as an expected result since, even though the interference state has a much higher probability of receiving packages with errors, the fact that the link has such a low probability of being in that state makes it so that the errors are usually registered while the link is in the normal state.

Task 1.b and c - probability of false positives and negatives

Matlab Code

```
for n = 2:5
    fprintf("\nN = %d\n", n)

    p_error_normal = 1 - (nchoosek(n_bits, i) * n_rate ^i * (1-n_rate)^(n_bits-i));
    p_error_normal = p_error_normal^n;
    p_error_interference = 1 - (nchoosek(n_bits, i) * i_rate ^i * (1-i_rate)^(n_bits-i));
    p_error_interference = p_error_interference^n;

    % probability of a false positive = p(being in the normal state |
    % every frame received with errors)
    p_false_positive = p_error_normal*probs./(p_error_normal*probs + p_error_interference*(1-
        probs))*100

    % probability of a false negative = p(being in interference state |
    % at least 1 frame is being received correctly, ie 1 - all frames with errors
    p_false_negative = (1-p_error_interference)*(1-probs)./(1-(p_error_normal*probs + p_error
        _interference*(1-probs)))*100
end
```

Code Analysis

For each n value from 2 to 5, we calculate the probability of existing, at least, one error in, at least, one package from the n control frames received for each state.
We then calculated the probability of being a false positive using Bayes' Law. We did a similar process for the false negatives only we had to use the complementary value of the one we calculated (*p_error_interference*).

Results - false positives

p	n = 2	n = 3	n = 4	n = 5
p = 99%	0.0003	0.0004	0.0006	0.0001
p = 99.9%	0.0025	0.0041	0.0065	0.0010
p = 99.99%	0.0255	0.0408	0.0651	0.0104
p = 99.999%	0.2545	0.4076	0.6510	0.1040
		* 1.0e-4	* 1.0e-8	*1.0e-11

Results - false negatives

p	n = 2	n = 3	n = 4	n = 5
p = 99%	0.5915	0.7385	0.8326	0.8927
p = 99.9%	0.0589	0.0737	0.0831	0.0892
p = 99.99%	0.0059	0.0074	0.0083	0.0089
p = 99.999%	0.0006	0.0007	0.0008	0.0009

Task 1.d - Influence of p and n values

False Positives

The results show that an increase on the value of p, probability of being in the normal state, increases the probability of false positives around 10 times. Since a false positive is the link being detected as being in an interference state when it should be detected as being in a normal state, the greater the probability of being in the normal state, the greater the probability of false positives.
The results also show that an increase on the value of n, number of control frames, means a drastically decrease in the probability of false positives, since with more control frames it's less likely to be in a state of interference.

False Negatives

The results show that an increase on the value of p, probability of being in the normal state, decreases the probability of false negatives around 10 times. Since a false negative is the link being detected as being in an normal state when it should be detected as being in an interference state, the greater the probability of being in the normal state, the smaller the probability of being in the interference state and therefore smaller the probability of false negatives.
The results also show that an increase on the value of n, number of control frames, means an increase in the probability of false negatives, since the increase in the number of control frames leads to more chances of a frame arriving without errors. Since it takes just one frame without errors for the link to be considered as in "normal state" (possible false negative), the increase in the amount of control frames to analyze reflect that.

Task 1.e

Having in mind the results, when aiming for a system where both false positive and false negative probabilities are not higher than 0.1%, the best value for n, number of control frames, when the probability of the normal state is p = 99.999% is n=3, since it's the smallest value when both probabilities are less than 0.1% (if n=2, the probability of false positives is 0.25%). Choosing the smallest value is important because more control frames will mean more bandwith of the link will be used for control and not frames with data.

Task 2

Task 2.a

Matlab Code

```
birthRate = [1 20 10 5];
deathRate = [180 40 20 2];

aux1 = birthRate ./ deathRate;
aux2 = [aux1(1) aux1(1)*aux1(2) aux1(1)*aux1(2)*aux1(3) aux1(1)*aux1(2)*aux1(3)*aux1(4)];

timePercentage0 = 1/(1+aux2(1)+aux2(2)+aux2(3)+aux2(4));
timePercentage1 = aux2(1)*timePercentage0;
timePercentage2 = aux2(2)*timePercentage0;
timePercentage3 = aux2(3)*timePercentage0;
timePercentage4 = aux2(4)*timePercentage0;
timePercentage = [timePercentage0 timePercentage1 timePercentage2 timePercentage3
    timePercentage4];
%timePercentage0+timePercentage1+timePercentage2+timePercentage3+timePercentage4
fprintf("2.a) %d\n", timePercentage)
```

Code Analysis

We began the same way we did in Task 1: defining constant values. In this case, we defined the birth and death rates for each state.
We then calculated the steady-state probability of the birth-death Markov chain. These values corresponded to the average percentage of time the link is on each of the five possible states.

Task 2.b

Matlab Code

```
errorRate = [10^-6 10^-5 10^-4 10^-3 10^-2];
linkAvgErrorRate = sum(timePercentage.*errorRate);
fprintf("2.b) %d\n", linkAvgErrorRate)
```

Code Analysis

Like in exercise 2.a, we began with constant value definition: *errorRate*. We multiplied each state error rate by its time percentage and added all the values. This resulted in the link average error rate.

Task 2.c

Matlab Code

```
avgHoursEachState = [1/birthRate(1) 1/(birthRate(2)+deathRate(1)) 1/(birthRate(3)+deathRate
    (2)) 1/(birthRate(4)+deathRate(3)) 1/deathRate(4)]; % In hours
avgMinutesEachState = avgHoursEachState.*60;
fprintf("2.c) %d\n", avgMinutesEachState)
```

Code Analysis

In order to calculate each state average time (in hours) we used its birth and/or death rate (the first state only has a birth rate while the last only has a death rate).
Having calculated each state average time in hours, we only had to multiply that value by 60 in order to get the value in minutes.

Task 2.d

Matlab Code

```
% States in which the link is in interference state: 4(10^-3) and 5(10^-2)
probInterferenceState = timePercentage(4) + timePercentage(5);
fprintf("2.d) %d\n", probInterferenceState)
```

Code Analysis

There are only 2 states in which the link is in interference state: states 4 and 5.
Therefore, the probability of the link being in interference state was just the sum of the steady-state probabilities (stored as *timePercentage*) for those states.

Task 2.e

Matlab Code

```
avgBitErrorRateInInterferenceState = (timePercentage(4)*errorRate(4)+timePercentage(5)*
    errorRate(5))/probInterferenceState;
fprintf("2.e) %d\n",avgBitErrorRateInInterferenceState)
```

Code Analysis

In order to obtain the average bit error rate in interference state we multiplied the steady-state probability of each of the interference states by its error rate, added them and divided the result by the probability of the link being in an interference state.

Task 2.f

Matlab Code

```
% States in which the link is in interference state: 4(10^-3) and 5(10^-2)
E4 = avgMinutesEachState(4);
E5 = avgMinutesEachState(5);
res = E4 * 4/5;

for i = 1:10
    res = ( ((i+1) * E4) + (i * E5) )  * (1/5)^i * (4/5) + res;
end

fprintf("2.f)")
disp(res)
```

Code Analysis

The average duration in an interference state is the sum of the times spent in states in which the link is considered in interference state (states 4 and 5).
The link may switch from state 3 to 4 and then back to 3, spending only (in average) the average time of state 4. However, it may also switch from state 4 to 5 (instead of returning to state 3) which will later return to state 4. This "loop" (switching from state 4 to 5 and back to 4) may repeat an unlimited amount of times. In order to calculate the average time the link is interference state we have to consider the first case (no loop, just switching from state 3 to 4 and right back to 3) and the cases in which we consider one or more loops.
To do so, we began by calculating the average duration of the first case and used a for loop in order to calculate the average time spent in interference state if there were i loops. These values were multiplied by the probability of their occurence (the first case has a probability of $20/25 = 4/5 = 80\%$ while having 1 loop has a probability of $1/5 * 4/5 = 4/25 = 16\%$ and so on).
We calculated for i ranging from 1 to 10 since the values after than would have such a low probability they would barely change the result (11 loop iterations would have a probability of (1/5)^11 * 4/5 = 2*10^-8 * 4/5 = 1.6*10^-8).

Task Results

```
2.a) 0.9870%    0.0055%    0.0027%    0.0014%    0.0034%
2.b) 3.695682e-05
2.c) 60 minutes    0.3 minutes    1.2 minutes    2.4 minutes    30 minutes
2.d) 4.797807e-03
2.e) 7.428571e-03
2.f) 10.5 minutes
```