Computer Networks — Assingment #4

October 22, 2025

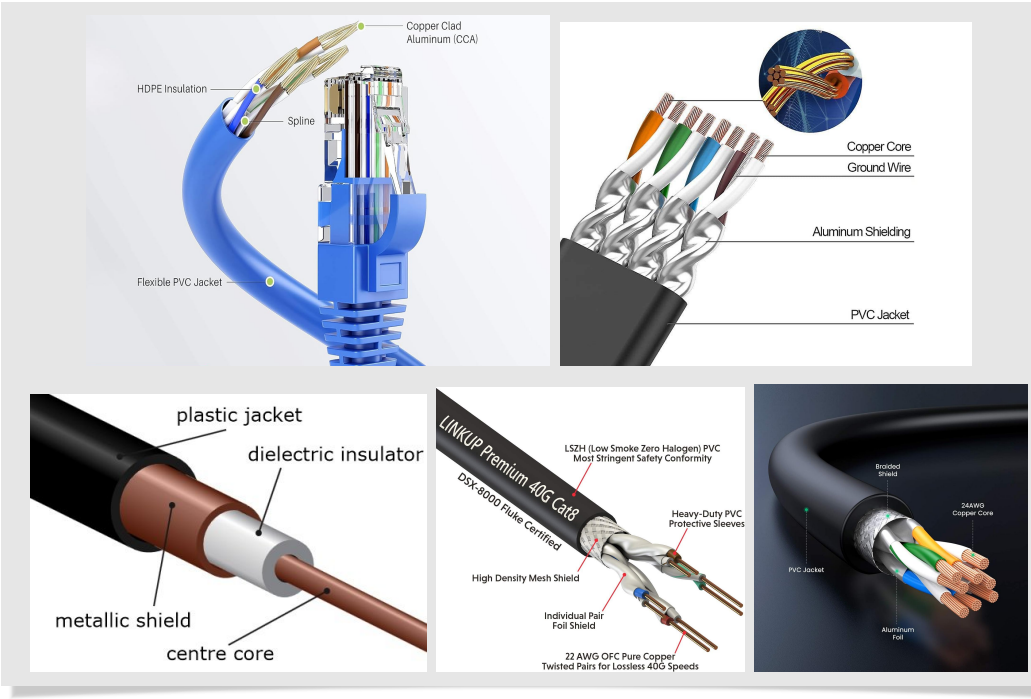
Assignment #4
Total: 8.5 points

*This assignment must be completed **individually**.*

I.e. you should work alone without any help or collaboration from others.

Please provide detailed answers to the following exercises.

NAME	
Student Nbr	



1. [1.25 pt]

Let A and B be two devices attempting to transmit on an Ethernet network. Each has a steady queue of frames ready to be send; A's frames will be numbered $A_1, A_2, ...$, and similarly for B's frames. Let $T = 51.2\mu$s be the exponential backoff base unit.

Suppose A and B simultaneously attempt to send frame 1, collide, and happen to choose backoff times of $0 \times T$ and $1 \times T$, respectively, meaning that A "wins" the race and transmits A_1 while B waits. At the end of this transmission, B will attempt to retransmit B_1 while A will attempt to transmit A_2. These first attempts will collide, but now A backs off for either $0 \times T$ or $1 \times T$, while B backs off for time equal to one of the $0 \times T, 1 \times T, ..., 3 \times T$.

a) Give the probability that A wins this second backoff race immediately after the first collision; that is, A's first choice of backoff time $k \times 51.2\mu$s is less than B's.

b) Suppose A wins this second backoff race. A transmits A_3, and when it is finished, A and B collide again as A tries to retransmit A_4 and B tries once more to transmit B_1. Give the probability that A wins this thriod backoff race immediately after this collision.

c) Obtain a reasonable lower bound for the probability that A wins all the remaining backoff races.

d) What then happens to the frame B_1?

This scenario is known as the Ethernet *capture effect*.

o

2. [1.25 pt]

Suppose Ethernet physical addresses are chosen at random (using true random bits):

(a) What is the probability that on a 1024-host network, two addresses will be the same?

(b) What is the probability that the above event will occur on some one or more of 2^{20} networks?

(c) What is the probability that of the 2^{30} hosts in all the networks of part b), some pair has the same address?

Hint: The calculation for parts a) and c), can be seen as a variation of the strategy used for the so-called "birthday problem": i.e. given N people, what is the probability that two of the birthdays (addresses) will be the same?

3. **[3 points]**

Start by considering 5 hosts that are waiting for another packet to finish on an Ethernet network. All the stations (hosts) transmit at once when the apcker is finished and collide.

(a) Start by simulating this situation by hand, ie. pretend that you have 5 hosts and reproduce the events described above until one of the five waiting hosts succeeds.
The main idea is that you will need to flip coins, roll a dice or some other genuine random source to determine backoff times.
Make the following simplifications: ignore interframe spacing, ignore variability in collision times (so that retransmission is always after an exact integral multiple of the 51.2μs slot time), and assume that each collision uses up exactly one slot.

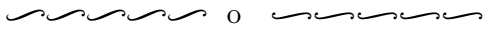
(b) Discuss the effect of these simplifications vs the behaviour one might encounter on a real Ethernet.

(c) Write a program that implements this simulation, but in this case that can handle N stations waiting to transmit.
Again, model time as integer T, in units of time slots or time steps, and also treat collisions as taking one slot time (so a collision at time T followed by a backoff of $k = 0$ would result in a retransmission at $T + 1$).
Find the average delay before one station transmits successfully, for $N = 20, 40$ and 100.
What trend do you notice with N? Does it make sense?

A few comments/suggestions:

- For each station, it would be a good idea to keep track of the station's "NextTimeToSend" and "CollisionCount".
The simulation would be done, when it has reach a time T for which there is only one station with NextTimeToSend==T.
If there is no such station, increment T.
If there are two or more, schedule the retransmission and try again.

- One possible way to implement this is using an OOP approach, this implies to identifying what could be a good candiate for an object, as well as, what propertoes and methods to define wiht it.



4. **[3 points]**

Consider the following Ethernet model:

- Transmission attempts are made at random times with an average spacing of λ slot times; specifically, the interval between consecutive attempts is an exponential random variable $x = -\lambda \log u$, where u is chosen randomly in the interval $0 \leq u \leq 1$.
- An attempt at time t results in a collision if there is another attempt in the range from $t-1$ to $t+1$, where t is measured in units of 51.2 μs slot time, otherwise the attempt succeeds.

(a) Write a program to simulate, for a given value of λ the average number of slot times needed before a successful transmission, usually referred as the *contention interval*. Find the minimum value of the contention interval. Note that you will have to find one attempt past the one that succeeds in order to determine if there was a collision. Ignore retransmissions, which probably would not fit in this random-based model anyway.

We have included for your ref. a plot, see Fig. 1, with the qualititave estimation of the behaviour you should observe for the value of the *contention interval* vs λ. **Notice though that in simulations like this, although the overall trend and final results are similar, the specific details and values are not due to the random nature of the implementation.**

(b) The Ethernet alternates between contention intervals and successful transmissions. Suppose the average transmission lasts 8 slot times (512 bytes). Using the minimum length of the contention interval determined before, calculate what fraction of the theoretical 10-Mbps bandwidth is available for transmissions?

You may choose the language of your preference to write the simulation program, e.g. C, C++, Fortran, or Python. Do not use, proprietary applications like matlab. Write good quality code, with proper documentation and modular. Do not hardcode values, instead use

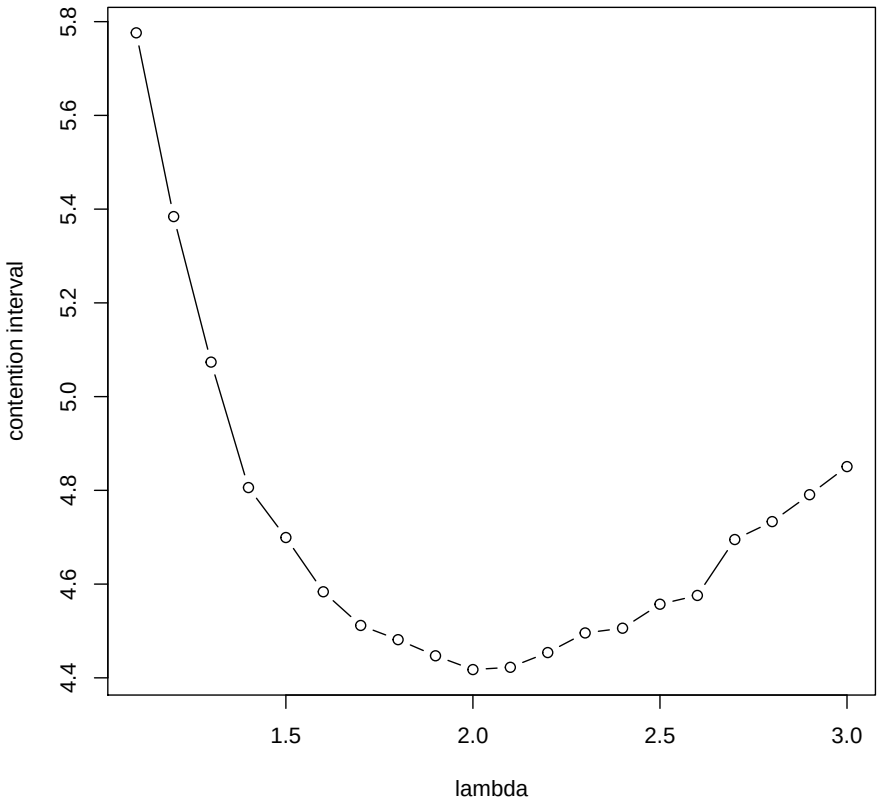


FIG. 1: Contention interval vs λ, in arbitrary units. Result from the model proposed in Problem #2.

parameters and/or arguments with default values, that can later be modified for representing different situations. Include details in how to compile and run your code. We should be able to "reproduce" your results at the qualitative level, and quantitatively by setting the seed of the *pseudo-random number generator* (PRNG).

Be really careful about the random number generation algorithm, use an appropriate one such as, Mersenne Twister [1].

o

[1] https://en.wikipedia.org/wiki/Mersenne_Twister