DISI – University of Trento

Master in Computer Science AA 2018/2019

Simulation and Performance Evaluation

Assignment 2

Simulating Aloha-like MAC Protocols in Python

Renato Lo Cigno, Michele Segata

May 20, 2019

We start from the simple simulator framework that is available through classroom and on the git repository. The assignment is to select one of the following extensions, design it (flow chart, modifications, ...) implement it, run proper simulations and write a report presenting and commenting the simulation results. The modified code must be delivered with the report, properly commented, with all the scripts needed to run it and to obtain the results presented in the report, including the scripts to analyze and plot results. The report should be concise and clear, typically no more than 4-5 pages in the given format. Report also the topology of your network and use it to comment how results differ for what you would expect from a theoretic analysis where all stations hear one another.

Not all modifications have the same complexity, some are really trivial. Thus we propose you different extensions, but some of them limit the maximum grade you can achieve: Less work, limited insight in the concepts of DES, limited grade: looks fair!

In any case each of you is assigned a network topology (see the `topologies.zip` file in Classroom to find your `<surname>.topo` topology description) that must be used to obtain results, so that even if some of you implement the same extension together results will be different and need a different explanation. The topology description is a simple text file which includes the `nodes` parameter to be copy-pasted inside your simulation configuration file (`config.json`), i.e., something like

```
"nodes" : [
    [[0,0], [1,2], [2,3], [3,4]]
],
```

To draw your topology, we provide you with an R script that you can use as follows:

```
Rscript draw-topology.R <surname>.csv
```

This produces `topology.pdf` showing you the position and the links between the nodes. Remember to pass the `csv` file you find in `topologies.zip`, not the `topo` file. The latter is for the simulator.

The inter-arrival times in the configuration file must be adapted to the number of nodes you have in the topology to fit a meaningful load range on the channel. Before importing in the simulator your topology, run the simulator with a simplified topology where the number of nodes you have been assigned can all hear each other, i.e., they all lie within a circle of 10 m as in the simulator we have defined, conventionally, 10 m to be the transmission/reception range, similar to sensor network devices following IEEE 802.15 standards that limit transmission power to 10 mW. Notice that our simulator works with an abstraction level where transmission details such as power and modulation are not needed. This basic result is very useful to present and comment results.

The following list reports the proposed modifications with the maximum grade available (31 means '*cum laude*'). You do not have to declare in advance what modification you choose, so you can also start with an easy one, then move on to a more complex task based on the simpler one. However,

you have to state explicitly the modification you choose in the report. If none of these modifications satisfy your curiosity or thirst of knowledge, feel free to propose another one, but do it before starting to implement and discuss it with us, to avoid entering a dead end or too complex affair.

No modification (Max: 22/30). Simply use the simulator and properly comment results comparing the difference due to the use of a realistic topology w.r.t. to the base case where all nodes hear one another. You can/should change some of the parameters of the simulator, like the buffer size at stations to get more interesting results.

Trivial Carrier Sensing (Max: 26/30). Extend the protocol to include a trivial carrier sensing function that prevents transmission if there are packets being transmitted within the communication range of the node. The packet is transmitted when the channel becomes free (1-persistent protocol). Use a sensing time $T_s = 50\,\mu$s.

Compare results with the original simulator.

Simple Carrier Sensing (Max: 27/30). As the **Trivial Carrier Sensing**, but a p-persistent version. A station before transmitting must sense the channel. If the channel is free, it proceeds to transmit; if it is occupied, it re-schedules the transmission with persistency p. p is a probability that defines if the transmission is scheduled immediately after the channel becomes free, or if it is re-scheduled after an exponential random time with average $T_r = 10T_t$, where T_t is the time needed to transmit the maximum size packet. If the transmission is scheduled immediately after the channel becomes free, no carrier sensing is performed and the packet is transmitted after the processing time, if there are other packets in transmission there is a collision. If the transmission is instead re-scheduled, carrier sensing is performed again before transmitting: if at the moment of transmission the channel is busy, the procedure is repeated.

Performance evaluation must include a sensitive analysis on p.

Carrier Sensing with Collision Avoidance (Max: 28/30). Extend the **Trivial Carrier Sensing** protocol, but including a collision avoidance procedure with a slotted contention window when the channel becomes free. The slot duration is T_s and the contention window size is W_c.

If the channel is busy, wait until it becomes free, then extract an integer random variable B_k uniformly distributed in $[0, W_c - 1]$ and count-down B_k slots. When $B_k = 0$ transmit; if the channel becomes occupied again, then re-schedule the transmission when the channel becomes free again extracting a new value for B_k. After a transmission the station must behave as if the channel were occupied.

Compare the results for different values of W_c. Set T_s to a value that is "justified" by all the other parameters of your protocol.

Realistic Propagation (Max. 27/30 or + 3 on any modification above). Change the packet reception model from a disk model to a probabilistic model where the probability of reception is inversely proportional to the distance. More formally, the reception probability for a packet so far is modeled as

$$\Pr(\text{correct reception}|d) = \begin{cases} 1 & \text{if } d < \text{RX}_{\text{range}} \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

In this extension, change the model to implement the following law

$$\Pr(\text{correct reception}|d) = \begin{cases} 1 - [\frac{d}{\text{RX}_{\text{range}}}]^{\frac{1}{3}} & \text{if } d < \text{RX}_{\text{range}} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

This models the reception probability when no collision occur. The probability of reception for colliding packets is still zero, and collisions should be properly separated from corrupted packets. A corrupted packet may still collide with a good one "destroying" it.

Unicast Traffic: (Max 31/30). Implement destinations and acknowledgment of packets, i.e., stations must be identifiable and packets are sent from one specific station to another one, which, upon correct reception, sends an acknowledgement back. Packets that are not acknowledged must be re-sent after an exponential random time with average $T_r = 10T_t$, where T_t is the time needed to transmit the maximum size packet. Two timing parameters must be defined: SIFS and DIFS (we take WiFi terminology here). SIFS (Short Inter-Frame Space) is $10\,\mu$s and separates a frame (packet) from its ACK. DIFS (Distributed Inter-Frame Space) is $50\,\mu$s and separates two different frames, or better, the end of a transmission (reception) from the attempt to send a new frame.

Carrier sensing must be implemented with $T_s =$ DIFS to protect ACKs from collisions. Collided packets are re-scheduled after $T_r = 10T_t$, where T_t is the time needed to transmit the maximum size packet.

802.11: (Max 31/30). Implement the full (well a bit simplified!!) MAC of WiFi as follows.

You have to start from the **Unicast Traffic** extension and change the channel access mechanism like in the **Carrier Sensing with Collision Avoidance**. The slot duration is $T_s = 20\,\mu$s, and the channel must be sensed free for an entire DIFS before starting to contend for the channel, i.e., before the transmission begins if the channel is free or the Collision Avoidance procedure is run.

When a packet is lost (the ACK is not received), a node should double its contention window for extracting the B_k value up to a maximum value $W_{\max}$. Let's suppose the contention window is set to the initial value $W_c = W_{\min}$. When a packet is lost, the node schedules the re-transmission and doubles the contention window size, i.e., $W_c = 2 \times W_c$. The "game" is repeated until the packet finally gets through or a maximum number of attempts N_a is reached. When successfully sending a packet (or when giving up after N_a attempts) the contention window is reset to the minimum before trying to send the next packet (i.e., $W_c = W_{\min}$ whenever a new packet is sent). In basic, standard WiFi we have $W_{\min} = 32$, $W_{\max} = 1024$, $N_a = 7$.

Further, when the channel turns busy while performing the countdown for channel access, the value of the countdown should be "frozen", and the node should continue with the countdown when the channel turns free again, without extracting a new value for B_k.

Notes on results and report preparation

The simulator uses a channel capacity at the MAC layer of $C = 8000000$ bit/s, it's just a convenient and reasonable value for the kind of MAC protocol we use. You can change it if you want, but be careful to check that everything remains coherent with your computations.

The goal of the simulations is to evaluate the throughput obtained, the collision rate, and the packet loss rate on station queues, i.e., when a packet arrives at a station and it cannot be stored into the transmission buffer. The throughput in the simulator is measured at each single receiver. Notice that this is different from the notion often used of "channel throughput," as one packet can be received by many receivers and the throughputs cannot be simply "summed" as you would get a throughput larger than the channel capacity. Even the average of the receiver's throughput is not exactly the channel throughput, which can instead be computed only when every station hear each other as the fraction of channel time occupied by packets correctly received by every station. The collisions again are measured at receivers; however, the interpretation of collision may be more delicate as packets colliding at one receiver may not collide at others: be careful not to obtain collision rates larger than 1!

The simulator logs events, a separated script manipulates the log to obtain performance measures.

The buffers at stations, currently set at 2, measure the number of packets that can be stored either while transmitting another or while receiving a packet. To have meaningful simulation results, the system must be at steady state, otherwise we measure a portion of a transient, which is normally very difficult to interpret. This is why we set the buffer so small: the longer the buffer the longer the

transient of the simulation before it reaches steady state, specially in the load are of greater interest. You are free to change it, but document it and analyze your system properly. Reasonable numbers in wireless network cards are anywhere between 1 and 100. If you set it at 0 you get ... very interesting results, try them and explain why.

The load on the channel is given by the amount of traffic generated by each station; the load can be modified changing the parameters of the distributions that describe the packets' size and inter-arrival time. The first thought of everyone is to plot throughput, collisions and losses as a function of this parameters ... which means that results are very difficult to understand and compare. Instead, you should compute the load (in bit/s) offered to the channel by all stations, normalize it w.r.t. C and then use this value ad the x axis of your plots.

If anything in this assignment is not clear, including the logic of the MAC protocols you should implement, **please contact us**. A simple misunderstanding might cause you to implement something completely wrong.

To write your report use the LaTeX template we give you and try not to write more than 4-5 pages. Deliver the PDF file of the report and all the code and scripts you used developed as a single .zip or .tar file through Classroom; if the simulator does not compile and run or the scripts do not run on a standard Linux box, we simply notify you that it does not work, and we will not attempt correction. Keep your code **CLEAN, ORGANIZED, and COMMENTED**. Do not send us your source code with unused portions commented out, blocks of code with no comments, or with monolithic pieces of code. Split your code in functions. DO NOT use absolute folders like

```
ds <- read.csv('/home/john.doe/Documents/spe/doe.csv')
```

but rather

```
ds <- read.csv('./doe.csv')
```

The deadline to deliver this assignment is July 31, 2019.
If you have some doubts, just write us an email or ask in class.

Have Fun!