David Colonna 18944827

Step 1

1.

- How would the channel capacity be measured ?

Since we only consider a network that operates one one frequency, the channel capacity would describe the maximum information rate that the channel can trasnmit, hence the width of the frequency it transmits on.

To demonstrate this we could choose the following scenario : 2 radio emitter-receiver, that all transmit on the frequency with value 802-844Hz and we assume that there is no noise and only two signal levels. Then we can use the Nyquist Boiseless Channel Bit Rate : Bit Rate $= 42 * 2 * log_2(2) = 84bps$.

In order to measure the channel capacity we need to send the maximum quantity of data in order to saturate the bandwidth. First start with insanely large amounts which will result in an error telling us that this information cann not be sent through this channel, then continue trying by reducing the size of the message step-by-step until the sending of the message is a success.

Depending on how we want to measure it, we may or may not need to take into account protocol overhead. If we want to only find out about the "real" channel capacity, meaning the capacity of useful data the channel can support, then we need to factor in the overhead, else it doesn't matter. In the first case we need to subtract from the total channel capacity the bits per second of transmission that are needed for the overhead. Its size depends on the information the overhead conveyes. We only need to worry about "one" overhead at a time since by having only one frequency channel, we can only have one transmitter emit at a time.

2.

- Protocol Overhead :

This describes the data that has to be sent along every message in a network to arrive at destination in a an orderly fashion. Depending on the protocol used, this overhead can be larger or smaller, for example in TCP it is quiet large (40 to 80 bytes per packet) but it guarantees that the data will be received by its destination, in the order that it was sent and without being corrupted. The ideal protocol overhead size would of course be 0% of the total message size, since then the communication would be the most efficient. However in practice this is impossible since in this case we could have absolutely no gurantee regarding the transmission of the message.

3.

- Implications for throughput performance and transmission delay with an increasing number of transmitting radio stations:

Since there is an increasing number of transmitting radio stations that operate at the same time, this will severly impact the performance of existing networks and stations. If more and more and stations emmit on the same frequencies (channel) that means a lower throughput for all stations, but the throughput performance of the whole network for the same channel will stay the same. Another consequence of this change will also be that the transmission delay for each station will augment considerably, while it will probably stay the same for the whole network.

In particular for CSMA/CA (IEEE 802.11) whose principal characteristic is that its nodes avoid collisions by waiting for all other nodes to be idle before sending its message : this will mean that the delay for an individual node will be much longer, and also for the whole network. Regarding throughput, it will be lower for the nodes since they will have to wait for all other nodes to be idle. Hence the throughput of the whole network will also be lower on average. Also regarding the ALOHA protocol, since the idea is that you send data and if there is a collision you send it again, the consequences will be: since the nodes can send whenever they want, the throughput will stay the same, as wil the throughput for the whole network. For the delay, the effect will be comparable to that of the CSMA protocol or even worse, since as soon as there is a collision the message will need to be resent and there is no other organisation regarding sending messages.

Step 2

- Regarding the broadcacsting address, changing its value has no visible effect on the jemula output, but this is normal since this is not something that is measured in jemula802. It seems that the broadcasting address only needs to be changed if the current address is already in use by another braodcasting system.
- The fragmentation threshhold is the frame length that requires packets to be broken up into two or more frames. If you set a lower value this can reduce packet collisions since a longer frame length means that the channel is occupied for a longer time span, hence the probability for collsions is higher. Looking through the threshold.csv and delay.csv files we say no changes in the values when playing with these values.
- The long retry limit parameter represents how many times a long frame which experienced a collision will try to be sent again before giving up. This value doesnt seem to change any throughput or delay values.
- The short retry limit parameter represents how many times a short frame which experienced a collision will try to be sent again before giving up. This value doesnt seem to change any throughput or delay values.
- The MAC address paramter is pretty self-explanatory. Changing its value, we can already see in the jemula visualization that it has an impact on the simulation. From the total_delay.csv file we can see that : changing the number of MAC addresses breaks the whole xml scenario.
- Max Receive Lifetime: this reprensents the time a given station waits for a packet to be received, i. e. when a packet is received, the station checks its lifetime, if it is bigger than the threshold, then the packet is discarded, else it is kept. Here we can see that when lowering this value, there are a lot less packets that are received by the stations. hence the acceptance policy becomes stricter. It works also the other way around, since augmenting that limit gets more packets to be accepted.
- The Max Transmit MSDULifetime parameter specifies the maximum time that the 802.11 station can spend transmitting all management protocol data unit fragments for a MSDU packet.
- The RTS Threshold paramter specifies a request to send function to control the access the station has to the transmission medium. It reprensents a maximum size a packet can have until the request to send protocll is used in the transmission. By augmenting the threshold, more packets can be send quicker, but also there are more collisions, hence resulting in a smaller throughput on average. However lowering the threshold too much is also not a always a good thging since then very small packets need to wait a lot even though they could have been sent without experiencing any collisions.

Step 3

We first set the requiored parameters in the xml scenario file.

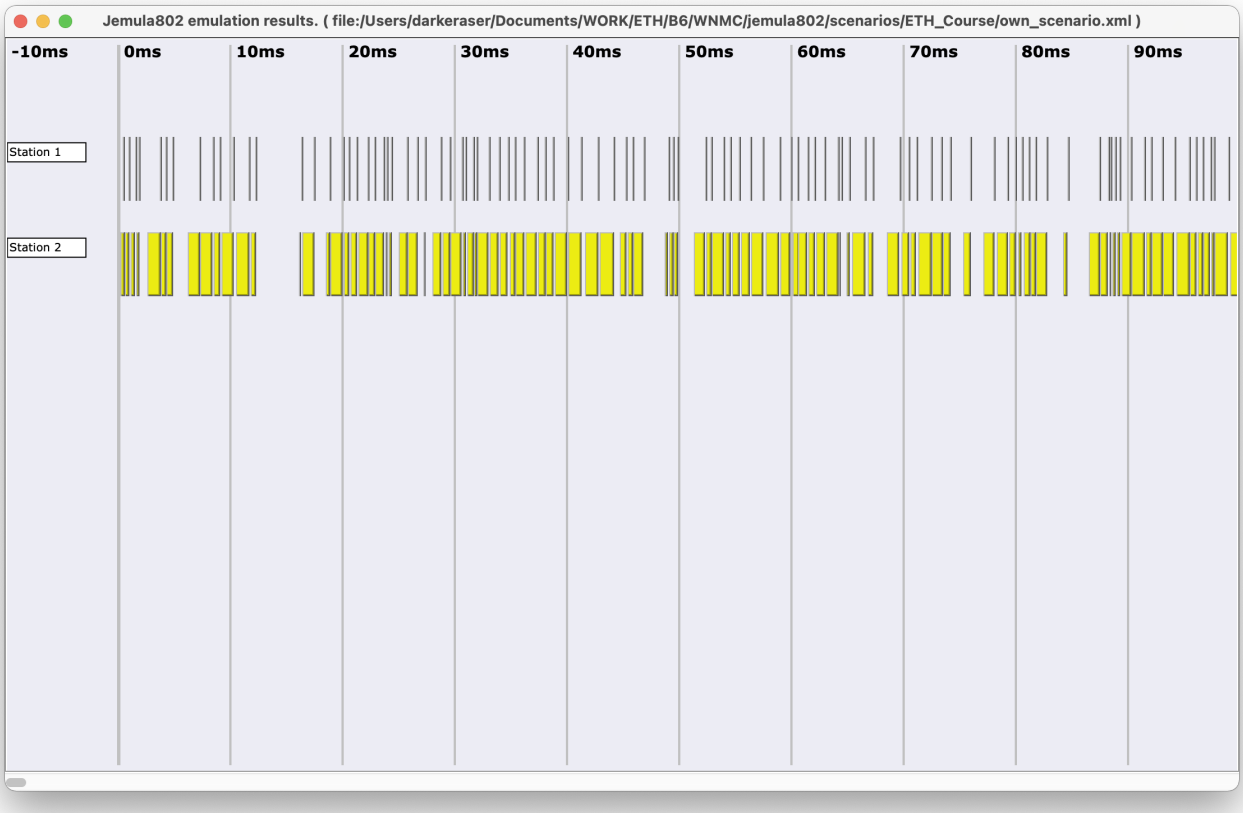
Step 4

Then we can see that by setting the mean load from 2.2Mbps to 4Mbps we get a significant better throughput (nearly twice the numbers of packets (680 to 1162)). However when we step up the mean load from 4 to 5 Mbps we only get a marginal throughput bump (+40 packets on average). Finally we see a maximum thoughput at 1200 on average when setting the mean load to 6 Mbps. Since when upping it to 7Mbps we get a worse throughput on average than before, hence

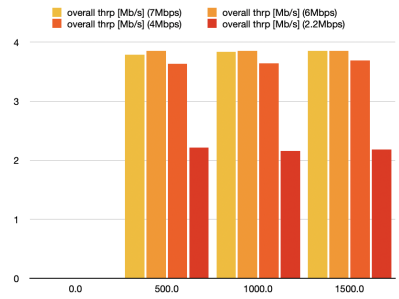
Gui Snapshot for 7Mbps mean load on 800 byte packets

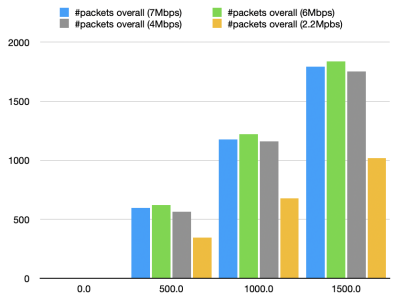
Gui Snapshot for 6Mbps mean load on 800 byte packets

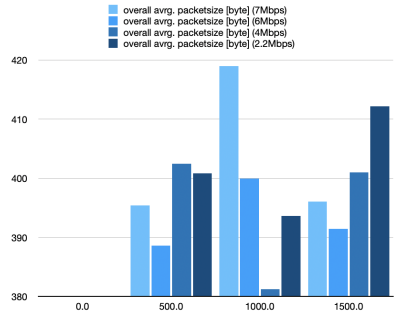
Gui Snapshot for 4Mbps mean load on 800 byte packets



Gui Snapshot for 2.2Mbps mean load on 800 byte packets


overall thrp [Mb/s] (7Mbps)
overall thrp [Mb/s] (6Mbps)
overall thrp [Mb/s] (4Mbps)
overall thrp [Mb/s] (2.2Mbps)
4
3
2
1
0
0.0
500.0
1000.0
1500.0


#packets overall (7Mbps)
#packets overall (6Mbps)
#packets overall (4Mbps)
#packets overall (2.2Mbps)
2000
1500
1000
500
0
0.0
500.0
1000.0
1500.0


overall avrg. packetsize [byte] (7Mbps)
overall avrg. packetsize [byte] (6Mbps)
overall avrg. packetsize [byte] (4Mbps)
overall avrg. packetsize [byte] (2.2Mbps)
420
410
400
390
380
0.0
500.0
1000.0
1500.0

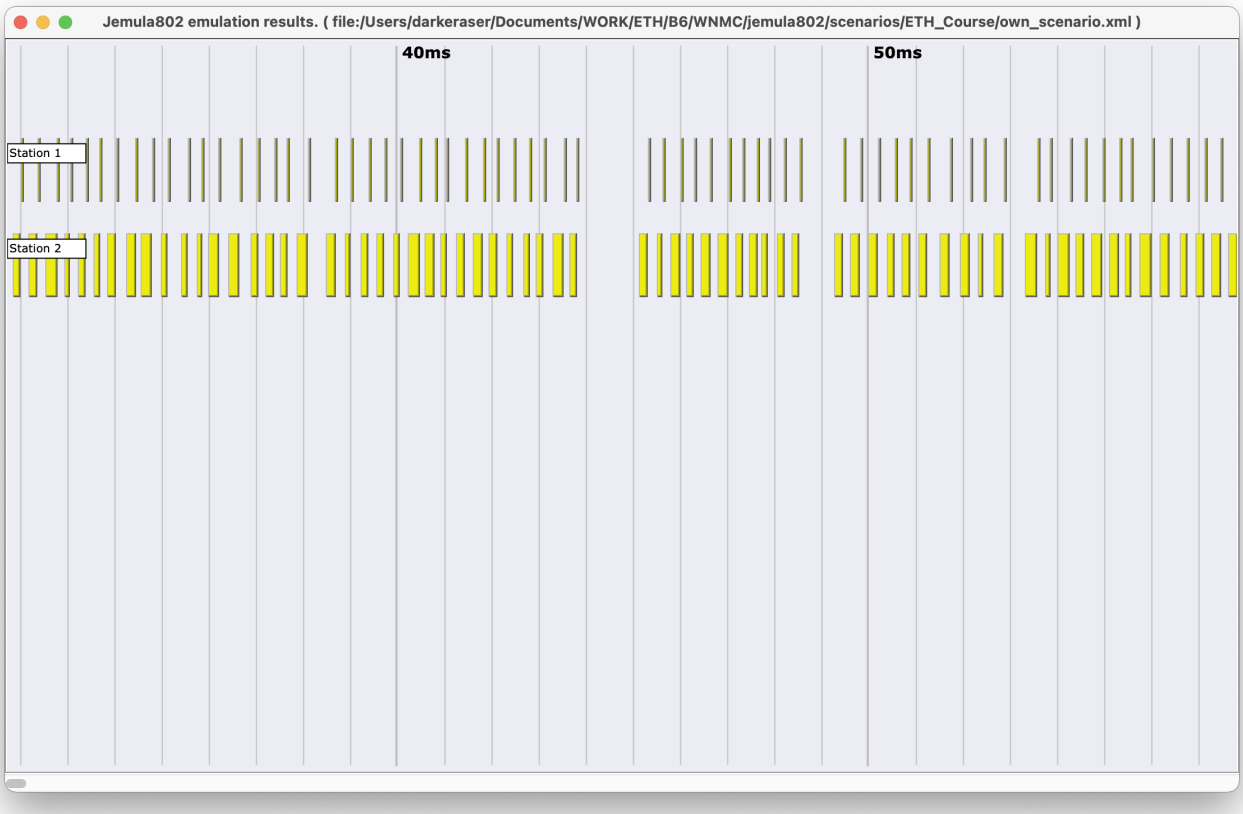
Motivation for RTS/CTS :

This mechanism is widely used in wireless networks to avoid packet collisions during data transmission by reserving the channel. The need that created this technique is called the **hidden node problem**. It can be described as follows : a node is called hidden when it can communicate with the networks access point, but it cannot communicate with the rest of the nodes. This means that in theory hidden nodes can send data simultaneaously to the access point resulting in interferences at the AP level and hence no packet comming through.

Step 5

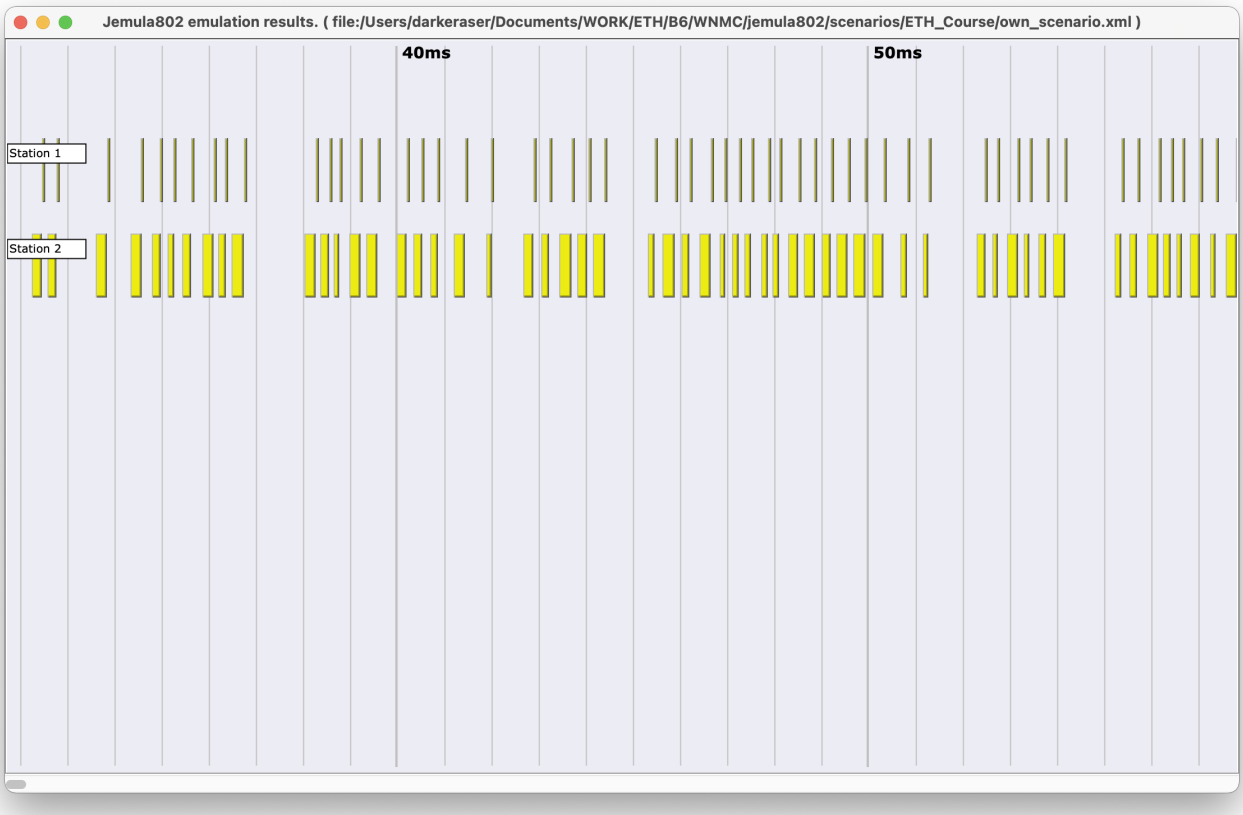
Packet Size of 120 bytes

Gui Snapshot for 2.2Mbps mean load on 120 byte packets



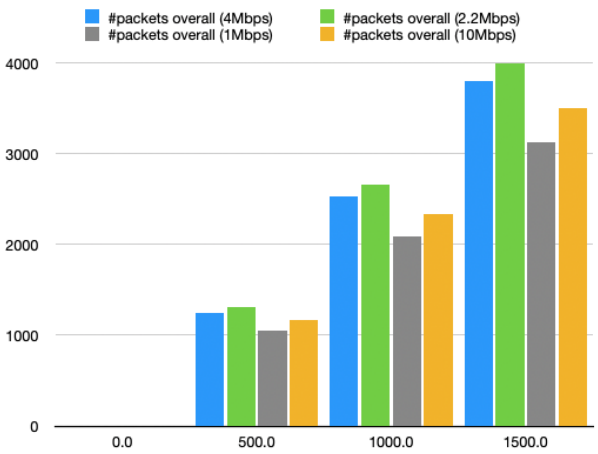
Gui Snapshot for 4Mbps mean load on 120 byte packets

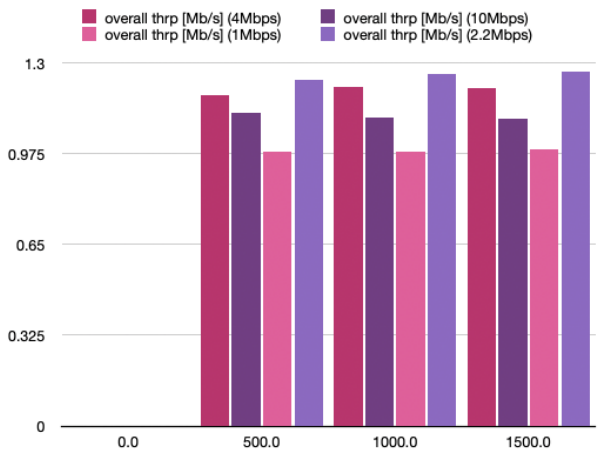
Gui Snapshot for 1Mbps mean load on 120 byte packets



Gui Snapshot for 10Mbps mean load on 120 byte packets


Jemula802 emulation results. (file:/Users/darkeraser/Documents/WORK/ETH/B6/WNMC/jemula802/scenarios/ETH_Course/own_scenario.xml)
-10ms
0ms
10ms
20ms
30ms
40ms
50ms
60ms
70ms
80ms
90ms
Station 1
Station 2


#packets overall (4Mbps)
#packets overall (1Mbps)
#packets overall (2.2Mbps)
#packets overall (10Mbps)
4000
3000
2000
1000
0
0.0
500.0
1000.0
1500.0


overall thrp [Mb/s] (4Mbps)
overall thrp [Mb/s] (1Mbps)
overall thrp [Mb/s] (10Mbps)
overall thrp [Mb/s] (2.2Mbps)
1.3
0.975
0.65
0.325
0
0.0
500.0
1000.0
1500.0

Interpretation

Contrarily to the previous packet size of 800 bytes we can see here that we don't really reach a maximum threshold until which the through put keeps getting bigger by augmenting the mean load. indeed we can see that the maximum throughput is reached for 2.2 Mbps mean load, and neither setting it to 1Mbps lower nor to 2 Mbps higher makes it higher. on the other hand we also observe that setting the mean load to 10Mbps doesn't drastically lower the throughput. It stays at the same level as for 2.2 Mbps mean load on average, which is interesting. As to why the performance changes, my guess would be that : since the packet size is much smaller than before, there can be more packets from hidden nodes sent to the AP hence creating more interference. Regarding the difference in number of packest sent overall, I think that this is because we have a much smaller packet size hence many more packets can be sent in the same time span.