Name/s: _____

NIA: _____ Group/s: _____

TDS Time Driven Simulation TDS

Solution Guideline

Note: The intention of the solution's guidelines is to show what it needs to be done and what it should be obtained and say. The intention is not to provide exact solutions, as the explanations can be provided in different ways. The solutions are not always provided with the requested parameters, but they show what the answer should provide and how to explain it. Revise well the parameters requested to provide the data and explanations requested.

In this assignment we study a time driven simulation that implements the slotted aloha protocol. This program will be extended in the following assignments hence it is very important to understand this initial version. This assignment only targets to understand the provided code.

The Aloha protocol is the simplest protocol for a shared media and it is defined with the following rules. If the user has a packet to send it sends it and waits for an acknowledge (ACK) from the receiver confirming the reception. The time waiting for the ACK is defined by a timer. If the ACK is received (before the times expiration), it deletes the packet and repeats the same process with the next packet in the queue if there is any. If the timer expires before receiving the ACK, it assumes the packet suffers collision (with other packets in the shared media) and starts the retransmission process. The retransmission requires waiting a random time before trying again to transmit the same packet (this avoids colliding again with the same packets). These are all rules defining the Aloha protocol. This definition and the simulation model is explained in the slotted-aloha slides provided with this handout [2].

The slotted aloha protocol follows the same rules except that it divides the time in units named slots, and the transmissions must always start at the beginning of the slot. This avoids collisions at the middle of the transmission (if packets are of the size of the slot) and hence the collisions are complete or there are successful transmissions. This doubles the efficiency of the Aloha protocol at the cost of the time synchronization mechanism.

The relevant processes to implement are:

1. Generate data packets at each station attached to the network
2. Transmit a data packet
3. Decide if a transmission has been successful or otherwise has collided with another packet
4. Decide the time to wait before retransmission after a collision
5. Answer the packet reception by the receiver sending and ACK packet
6. Receive the ACK

This document is divided in four parts: Part I reviews how this simulation could be implemented in an event-driven model; Part II gets your familiar with the protocol with manual inspection of the output traces; Part III targets the understanding of the code at the structure level; Part IV targets the comprehension of the event-driven and time-driven simulation concepts comparing code aspects of the supermarket and aloha simulations. The last part is to summarize the learning process.

PART I: EDS Analysis (not to do for pre-submission)

Before looking at the time oriented simulation code, think how you would design the simulation of the slotted-aloha protocol with an event driven simulation applying the lessons learned from the previous EDS practice.

Question 1: How should the protocol be implemented in an event driven simulation? What events and actions must be defined?

Answer:

Event	Actions to take in each event
Packet ARRIVAL	If we are in an IDLE stateSi estem en estat IDLE generem un TRANSMETRE paquet. En qualsevol altre estat posem el paquet a la cua perquè aquesta estació està a l'espera d'alguna cosa i no pot transmetre. Es crea un altre event ARRIBADA per quan cal crear el següent paquet. Creem un event/temporitzador COLLISIO que indica el temps que cal esperar per considerar que hi ha hagut col·lisió.
TRANSMETRE paquet	envia el paquet marcant al canal la seva informació de transmissió i passar a estat ACK. Si hi ha alguna altra transmissió en el mateix moment crear event col·lisió al final de la transmissió.
COLLISIO	Passem a l'estat de resolució. Cal aplicar l'algorisme de resolució de col·lisions. Creem un temps d'espera abans de tornar a intentar i creem event/temporització TRANSMETRE paquet a aquell punt que representa la (re)transmissió
ACK	Es rep missatge ACK, es dona per bona la transmissió i per tant s'elimina el paquet de la cua i passem a estat IDLE. Si hi ha algun paquet a la cua, creem event TRANSMETRE per enviar el primer paquet de la cua.
COMENÇAR	Decidir l'hora de la primera arribada i crear l'event d'ARRIBADA d'aquest paquet
ACABAR	Vàries opcions: 1) acabar immediatament i imprimir els resultats 2) no deixar crear més events i acabar amb tot el procés dels events en curs (fins que no en creen més i la cua d'events estigui buida)

En cada event part de les accions és sempre actualitzar les estadístiques que calgui per monitoritzar les funcionalitats a estudiar. Com hem de comptabilitzar col·lisions crearíem l'event col·lisió.

També podríem cridar directament event TRANSMETRE si sabem acumular les accions i no cal actualitzar quan toca.... Amb això cal poder calcular el temps d'espera de resolució al moment del TRANSMETRE. Es pot fer de moltes maneres....

Question 2: How would the time Advanced in the event driven simulation?

Answer: Com qualsevol simulació orientada a events, segons els temps del events generats que pel cas hem proposat 6 tipus d'events en la pregunta anterior.

Si necessitem que passi alguna cosa a un temps determinat cal assegurar que hi hagi un event perquè el temps es pari a executar les accions.

PART II: Execution of the s-aloha code. (start pre-submission)

This sections focus on understanding the use of the code, understanding what values to provide to the input parameters and what configuration make sense and knowing how to interpret the output of the simulation. The output is numeric values that measure particular metrics and also traces to follow the protocol operation. The understanding at the trace level allows us to understand the implemented model of the slotted-aloha protocol without looking at the code. Looking at the traces we see what it does without knowing how it is done that. The how is studied in the following section when we get inside the code.

Execute the program with the provided input file **in-ref** without changing any parameter inside the code and taking the output to a file named **out**. The general command line execution is: **saloha <input_filename> <outpu_filename>**. The particular one for this execution is then: saloha in-ref out. If you use netbeans to run it, you have to configure the run command in the project configuration tab indicating the two command line parameters: in-ref out. If the configuration is correct, the program creates the out file with the same information of the out sample file provided with the handout. If the out file is not created it means that either the compilation or the execution was not correct or configured correctly.

Question 3: How long has the program taken to run? Include the out file information that helps determine this information and explain it.

Answer:

```
*************** START EXECUTION ******************
  Time : Wed Jun 21 12:30:21 2021
**************************************************
 NETWORK CONFIGURATION ---
    Number of stations                    :        10
    Slot Size (bytes)                     :       100
    Transmission Rate (Mbps)              :    100.00
    Contention Resolution Algorithm       :         D (D-deterministic, P pPersistence, B BEB, O Optimal
pPersistence)
    Probability of p-persistent protocol  :    0.0150 (only for fixed p-persistence)
TRAFFIC GENERATOR -------
    Normalized offered load               :  0.320000 (   32.0000 Mbps)
    Interarrival Distribution (E)Exp.     :         E
SIMULATION PARAMETERS ---
    Length of simulation in miliseconds   :     50.00 (      6250 slots)
    Start statistics at milisecond        :     30.00 (      3750 slots)
    Seed value (random = 0)               :      3412
    Significance level (alpha)            :  0.050000
    Z value of 0.975 (1-alpha/2)          :      1.96
    Target CI accuracy r in % [0..100]    :      2.00

End of input data---
Initializing......

Seed for traffic generator : 1325115094
 ________________________________________________


SUMMARY OF SIMULATION MEASURES ----------------
Total offered load                        : 0.323600
Utilization                               : 0.322400

*************** END EXECUTION ******************
  Time : Wed Jun 21 12:30:21 2021
**************************************************

finished!!!!!!!!!!
```

Al principi i final del fitxer out s'indica l'hora d'inici i final de l'execució. La diferència és el temps que ha tardat en executar-se. Tarda tan poc que no es diferencia. Quan anem afegint coses tardarà més.

També a la finestra output el compilador aporta el temps d'execució al acabar.... imprimeix un text com:

RUN SUCCESSFUL (total time: 125ms)

Però aquesta dada surt per pantalla i no es guarda al fitxer output.

Question 4: What simulation duration (in ms) is requested (Length of the simulation line) in the run looking at the Length of the simulation line? Why this time is not the same as the simulation duration time in the previous question? Justify your answers.

Answer: The simulation duration is 300 msec. This is the time it simulates and other times provided by the output are: the real time the packets would take to be transmitted, the timers duration … However, this scenario model needs to do few operations and hence the time it takes to run these operations (logical time) is much faster than the real time they would take place in the real world (real time).

Question 5: Understand how to modify the input parameters to execution the program with a different configuration scenario. Execute the program again but now with the provided *in-ref* file and modify the parameters to set the values indicated in the table below. Fill the table with the output results obtained with each execution:

Answer:

Number of Stations	Simulation length	Offered load	Utilization	Execution Duration
10	300	0.321807	0.322222	0
10	50	0.3236000	0.322400	0
50	300	0.318667	0.318756	0
50	50	0.333200	0.335200	0

Question 6: Modify in the saloha.h file the variables used to generate DEBUG traces. The traces show the protocol operations in a concise form but detailed enough to follow all information requested. The actual wording makes easy to interpret provided information. Analyze the output traces and respond the following questions. If you need help in the meaning of parameters or variables provided you can check the comments on the variables definition inside the saloha.h file.

Some general guidelines on the trace format to interpret them. All traces follow a general format defined as : <time_slots> <num_station> <message> on

- time_slots : is the current simulation time that occurs this line of the trace
- num_station: station number that creates the message
- message: is the description of the operation executed at this time so that to follow all elements of the protocol. In capital there is always the keyword of the message that allows identifying the message among all lines in the trace. It is followed by the relevant information of this operation needed to follow the functionality expected.

6.a) Configuration DEBUGTRAF = 1, all the rest as in the original in-ref file

6.a) 1.- What station is the first one to generate a packet? What information you use to know it? Include the trace lines you use to identify this operation and explain them (Hint: check the messages INIT TRAF or PK ARRIVAL)

Answer:

```
0 STN  4 INIT TRAF: Next pak Arrival 3.247121 slots 4
```

El missatge INIT TRAF: més petit és l'estació 4 que dona un paquet arrival al temps 3.247121 per tant l'arribada afectiva es realitza a l'eslot 4. Efectivament el primer PK ARRIVAL que s'imprimeix a la traça és:

```
   4 STN  4 PK  ARRIVAL: pk (num   0, arv  3.247121 sarv    4 Tx-count  0) QU Queue (H  0,T  0, L  1)
(NM,SAV, iST,TX):   0 (  0,    4,   -1, 0)
```

A l'eslot 4 (primer número de la traça) de l'estació 4 i s'afegeix un element a la cua.

6.a) 2.- How many times has this first packet been transmitted to achieve the successful transmission? Which slot time is this packet successfully transmitted? Include the trace lines that help identify this information and explain them.

Answer: The first packet is successfully transmitted the first time (no retransmissions are needed) and this transmission occurs in slot 4 at the same time the packet arrives. The line that shows this information is:

```
4 STN  4 TRANSMIT : SA  4 DA 10 pk   0 channel state  1 stn state (prev I next T) attempts 1
```

This line indicates that station 4 ha has transmitted packet 0 having in the channel just this packet (channel state 1) and this is the first attempt (attempts 1).

Note that this line alone just shows that the packet is transmitted. But the trace does not have any other TRANSMIT message at slot 4. Hence this means it is a successful transmission with no collision.

6.a) 3.- What is the packet number of the last packet generated by station 8 and when (slot number) is it generated? What time should the next packet be generated and why it is not generated? Justify the explanation including the relevant trace lines and explaining them.

Answer:

6234 STN 8 PK ARRIVAL: pk (num 207, arv 6233.409750 sarv 6234 Tx-count 0) QU Queue (H 0,T 0, L 1) (NM,SAV, iST,TX): 0 (207, 6234, -1, 0)
6234 STN 8 NEXT INTARV: next-pk 208 Next time 6233.4097 ms 779177 ia 24.340628
6234 STN 8 NEXT PK ARV: next-pk 208 Next time 6257.7504 ms 782219

El últim PK ARRIVAL de l'estació 8 és a l'slot 6234 i és el número de paquet 207. El següent paquet 208 s'ha de generar l'slot 6258 que està fora de la durada de la simulació que és 6250 slots i per això ja no es genera.

6.a) 4.- Which is the packet that has required more retransmissions to be successfully transmitted among all packets transmitted during the entire simulation duration? How many times were needed? Indicate which station has generated the packet, when the packet is generated, and when it has been transmitted? How many packets the station has waiting for transmission in the queue at the time of successful transmission of this packet (Hint: check the highest value of the *attempts* `parameter in the trace). Include all relevant trace lines with the explanation.

Answer: El valor màxim d'intents de transmissió (attempts) es mostra al següent missatge:

4514 STN 2 TRANSMIT : SA 2 DA 10 pk 146 channel state 1 stn state (prev R next T) attempts 18

Que indica que aquest paquet s'ha transmès satisfactòriament ara slot 4514, és de l'estació 2 i és el paquet 146 d'aquesta estació i s'ha transmès 18 vegades (o sigui 17 col.lisions i la transmissió en èxit). Abans d'aquest missatge podem mirar el PK ARRIVAL previ d'aquesta estació que és:

4450 STN 2 PK ARRIVAL: pk (num 147, arv 4449.544829 sarv 4450 Tx-count 0) QU Queue (H 0,T 1, L 2) (NM,SAV, iST,TX): 0 (146, 4446, 4446, 1)
1 (147, 4450, -1, 0)

Això indica que la cua té 2 elements (Length L 2) i són els paquets 146, 147

El paquet 146 s'ha generat al slot 4446 ja que podem veure PK ARRIVAL:

4446 STN 2 PK ARRIVAL: pk (num 146, arv 4445.674119 sarv 4446 Tx-count 0) QU Queue (H 0,T 0, L 1) (NM,SAV, iST,TX): 0 (146, 4446, -1, 0)

6.a) 5.- Which is the slot that collide more packets at the same time? If there is more than one explain the one that happens first. (Hint: Identify the maximum value in *channel state*) Which stations and packets are involved in this collision? How many times have been transmitted these packets at the time of this collision? And how many times each of these packets need to be transmitted to have the successful transmission and at what slot each of this transmissions occur? Include all relevant trace lines with the explanation.

Answer: A l'slot 4893 hi ha 5 transmissions de les estacions 2,4,5,7,8 dels seus paquets 157, 139, 166, 140, 159 i aquest és l'intent 1,9,1,10,2 de cada un dels paquets respectivament com es pot veure en les línies següents:

4893 STN 2 TRANSMIT : SA 2 DA 10 pk 157 channel state 1 stn state (prev I next T) attempts 1
4893 STN 4 TRANSMIT : SA 4 DA 10 pk 139 channel state 2 stn state (prev R next T) attempts 9
4893 STN 5 TRANSMIT : SA 5 DA 10 pk 166 channel state 3 stn state (prev I next T) attempts 1
4893 STN 7 TRANSMIT : SA 7 DA 10 pk 140 channel state 4 stn state (prev R next T) attempts 10
4893 STN 8 TRANSMIT : SA 8 DA 10 pk 159 channel state 5 stn state (prev R next T) attempts 2

Aquests paquets s'acaben transmeten a:
4897 STN 2 TRANSMIT : SA 2 DA 10 pk 157 channel state 1 stn state (prev R next T) attempts 2
4899 STN 4 TRANSMIT : SA 4 DA 10 pk 139 channel state 1 stn state (prev R next T) attempts 10
4914 STN 5 TRANSMIT : SA 5 DA 10 pk 166 channel state 1 stn state (prev R next T) attempts 4
4902 STN 7 TRANSMIT : SA 7 DA 10 pk 140 channel state 1 stn state (prev R next T) attempts 11
4923 STN 8 TRANSMIT : SA 8 DA 10 pk 159 channel state 1 stn state (prev R next T) attempts 5

Així tenim que el paquet 157 de l'estació 2 s'ha transmès a l'slot 4897 a l'intent 2, els paquests 139, 166, 140, i 159, de les estacions 4,5,7, 8 s'han transmès a l'slot 4899, ,4914, 4902, 4923 a l'intent 10, 4, 11, i 5 respectivament.
Aquetes línies han estat escollides perquè el channel State en aquest slot es queda 1 al final de l'slot (no hi ha més transmits a aquest slot) i és l'últim transmit d'aquest paquet d'aquesta estació.

6.a) 6.- So far we have seen that with manual inspection of the traces we can follow how the program works and shows the detail of the particular operations executed. Do you think the information provided in the traces is enough to check the correctness of the **packet generation** process? Justify your reasoning using pieces of traces to show the exact verifications that can be done and which ones are missing if any.

Answer: Amb tot el què hem analitzat fins ara de la generació de paquets podem comprovar quan es decideix generar el següent paquet i a quin hora, realment el paquet es genera aquesta hora, se li assigna un número de paquet correcte i s'afegeix a la cua correctament. El què no hem mirat és si el temps entre arribades està ben calculat/implementat. Això ho mostra el missatge NEXT INTARV amb el número ia (interarrival):

458 STN 8 PK ARRIVAL: pk (num 18, arv 457.991648 sarv 458 Tx-count 0) QU Queue (H 0,T 0, L 1) (NM,SAV, iST,TX): 0 (18, 458, -1, 0)
458 STN 8 NEXT INTARV: next-pk 19 Next time 457.9916 ms 57249 ia 24.607798
458 STN 8 NEXT PK ARV: next-pk 19 Next time 482.5994 ms 60325

Veiem que es genera el paquet 18 de l'estació 8 i el càlcul del següent temps a generar és el temps actual amb decimals (457.9916) li suma el valor obtingut entre arribades 24.607798 que dona 482.5994. És a dir el següent paquet es genera 483. I es pot comprovar que efectivament és així, ja que al slot 483 tenim el PK ARRIVAL del paquet 19 de l'estació 8:

483 STN 8 PK ARRIVAL: pk (num 19, arv 482.599447 sarv 483 Tx-count 0) QU Queue (H 0,T 0, L 1) (NM,SAV, iST,TX): 0 (19, 483, -1, 0)

6.a) 7.- Can this same trace be used to know and exactly verify the **queue** operations? This means, can we exactly verify that the packets are correctly stored in the queue while waiting for its successful transmission and correctly eliminated when the transmission is confirmed successfully? Justify your reasoning using pieces of traces to show the exact verifications that can be done and which ones are missing if any.

Answer: En aquest cas hem de mirar la cua com evoluciona en els moments que es modifica i assegurar que es modifica bé, i garantir que no es modifica quan toqui. Per això quan es modifica cal relacionar-ho amb l'estat anterior i assegurar que és tot correcte. Cal comprovar:

- Que s'afegeix un paquet correctament quan la cua està buida
- Que s'afegeix un paquet correctament quan la cua té elements
- Que s'elimina un paquet correctament quan la cua té més elements i no es queda buida
- Que s'elimina un paquet correctament quan la cua només té aquest element i no es queda buida

Les arribades és quan es pot posar el paquet a la cua, per això amb aquest missatge mostra l'estat i elements de la cua. Exemple: QU Queue (H 0,T 0, L 1) (NM,SAV, iST,TX): 0 (192, 6111, -1, 0)
Això indica primer element de la cua (Head), últim element (Tail), Longitud (L) de la cua i per cada element en la cua indica posició en el vector, (Numero paquet, slot d'arribada, inici del servei, i cops que s'ha hagut d'enviar)

Cal tenir en compte que hi ha una cua separada per cada estació.

Amb això podem veure si realment les operacions de cua són correctes.

- Que s'afegeix un paquet correctament quan la cua està buida
  - Cas primer paquet per exemple: 47 STN 9 PK ARRIVAL: pk (num 0, arv 46.472838 sarv 47 Tx-count 0) QU Queue (H 0,T 0, L 1) (NM,SAV, iST,TX): 0 (0, 47, -1, 0)
- Que s'afegeix un paquet correctament quan la cua té elements
  - El primer cop que la cua de l'estació 9 té 2 elements és: 321 STN 9 PK ARRIVAL: pk (num 7, arv 320.016898 sarv 321 Tx-count 0) QU Queue (H 0,T 1, L 2) (NM,SAV, iST,TX): 0 (6, 316, 316, 1) 1 (7, 321, -1, 0)
  - Veiem que hi ha dos paquets (num 6 i 7) i estan posats en ordre d'arribada. El segon és el generat en aquesta arribada, i el primer en la cua ja hi era i s'havia afegit al 316: 316 STN 9 PK ARRIVAL: pk (num 6, arv 315.722587 sarv 316 Tx-count 0) QU Queue (H 0,T 0, L 1) (NM,SAV, iST,TX): 0 (6, 316, -1, 0)
  - Per tant el segon s'ha afegit bé ja que la cua té longitud 2, head 0 i tail 1, i els dos paquets els visualitza correctament.
- Que s'elimina un paquet correctament quan la cua té més elements i no es queda buida
  - El paquet 6 s'elimina a l'slot 327 : 327 STN 9 TRANSMIT : SA 9 DA 10 pk 6 channel state 1 stn state (prev R next T) attempts 2
  - No visualitza com queda la cua
- Que s'elimina un paquet correctament quan la cua només té aquest element i es queda buida
  - El paquet 7 s'elimina a: 328 STN 9 TRANSMIT : SA 9 DA 10 pk 7 channel state 1 stn state (prev I next T) attempts 1
  - No visualitza com queda la cua

- Quan elimina no visualitza la cua com queda per tant només podem veure en que realment els paquets es transmeten i valorar quan torna entrar un paquet a la cua la cua està en la situació correcte.
    - El paquet 9 arriba a: 420 STN 9 PK ARRIVAL: pk (num 9, arv 419.447105 sarv 420 Tx-count 0) QU Queue (H 0,T 0, L 1) (NM,SAV, iST,TX): 0 (9, 420, -1, 0)
    - Aquí veiem que la cua és buida i és correcte. Podríem buscar un altre cas en què tenim una arribada sense que s'hagi quedat buit i per tant visualitzar quins paquets encara hi ha a la cua i s'afegeix. Exemple en el paquet 23, arriba sense elements a la cua, i a partir de llavors arriben correlativament els paquets 24, 25, i 26 que tots estan a la cua fins que el 23 es transmet a l'slot 892. Llavors no veiem que els paquets 24, 25, i 16 continuen a la cua. Però a l'slot 899 arriba el paquet 27 i llavors sí es visualitza la cua i es pot comprovar que és correcte perquè conté en ordre correcte els paquets 24, 25, 26 i 27. Per tant les altes i baixes funcionen.

Podem dir que encara que no visualitza tots els detalls mirant detingudament la traça tenim informació suficient per considerar que les operacions sobre la cua són correctes.

6.a) 8.- Can we know with this trace if a packet suffers **collision**, and how many times and can we precisely verify the correctness of the full mechanism involved in this process? Include the exact lines of the trace to explain the exact detail verifications it can be done and indicate which ones cannot be done if any.

Answer: Si perquè es visualitza quan un paquet es transmet amb el missatge TRANSMIT donant tota la informació sobre totes les transmissions d'aquest paquet. La transmissió actual està en channel state que si és 1 vol dir que només ha transmet l'estació i per tant ha tingut èxit, si és >1 vol dir vàries transmissions i per tant col·lisió. A més attempts indica quantes transmissions porta acumulades incloent l'actual. Els exemples concrets dels dos casos de traces es veuen a continuació:

- Exemple col·lisió: `892 STN 9 TRANSMIT : SA 9 DA 10 pk 23 channel state 2 stn state (prev R next T) attempts 2`
    - Channel state = 2 -> col·lisió
    - Attempts 2 indica segon intent
    - Efectivament veiem l'altre transmissió en el mateix slot 892 de l'estació 5: `892 STN 5 TRANSMIT : SA 5 DA 10 pk 33 channel state 1 stn state (prev I next T) attempts 1`
- Exemple transmissió correcte: `958 STN 9 TRANSMIT : SA 9 DA 10 pk 23 channel state 1 stn state (prev R next T) attempts 8`
    - Channel state = 1 -> collisió
    - Attempts 8 indica vuitè intent del paquet 23 per transmetre's en èxit

6.a) 9.- Can we know with this trace if the **CRA algorithm** that computes the waiting time before retransmission works correctly? Include the exact lines of the trace to explain the exact detail verifications it can be done and indicate which ones cannot be done if any.

Answer: Com s'està aplicant l'algorisme deterministic que el temps d'espera és igual que el número d'estació podem comprovar que realment és correcte. Una col·lisió a l'estació 4 (amb l'stació 9) passa a l'slot 85 (intentant transmetre per primera vegada el paquet 0). A l'slot 86 es detecta la col·lisió (que en el mon real seria més perquè caldria posar un temporitzador, aquí assumim només el temps de propagació del paquet). Decideix quan temps ha d'esperar immediatament després per tant al següent slot 87 que li dona esperar 4 slots que aquest slot 87 ja és un de compte enrere. Per tant la següent retransmissió passa a l'slot 91 que és quan el compte enrere dona 0. En aquest cas no hi ha col·lisió i es rep el ACK el mateix 91.

```
85 STN  4 TRANSMIT : SA  4 DA 10 pk   0 channel state  1 stn state (prev I next T) attempts 1
85 STN  9 TRANSMIT : SA  9 DA 10 pk   3 channel state  2 stn state (prev I next T) attempts 1
86 STN  4 CRA DETERMINISTIC: wait  4
86 STN  4 COLLISION: SA -1 DA -1 pk   0 channel state  0 stn state (prev T next R) attempts 1 wait  4
86 STN  9 CRA DETERMINISTIC: wait  9
86 STN  9 COLLISION: SA -1 DA -1 pk   3 channel state  0 stn state (prev T next R) attempts 1 wait  9
87 STN  4 CRA: wait  4
87 STN  9 CRA: wait  9
88 STN  0 PK  ARRIVAL: pk (num   3, arv 87.709232 sarv   88 Tx-count  0) QU Queue (H  0,T  0, L  1) (NM,SAV, iST,TX):   0 (  3,   88,   -1, 0)
88 STN  0 NEXT INTARV: next-pk 4 Next time  87.7092 ms  10964 ia 0.681229
88 STN  0 NEXT PK ARV: next-pk 4 Next time  88.3905 ms  11049
88 STN  0 TRANSMIT : SA  0 DA 10 pk   3 channel state  1 stn state (prev I next T) attempts 1
88 STN  4 CRA: wait  3
88 STN  9 CRA: wait  8
88 STN  0 RV ACK   : SA  0 DA 10 channel state  1 stn state (prev T next I) tx-count 1 Queue (H -1,T -1, L  0) (NM,SAV, iST,TX):
89 STN  0 PK  ARRIVAL: pk (num   4, arv 88.390461 sarv   89 Tx-count  0) QU Queue (H  0,T  0, L  1) (NM,SAV, iST,TX):   0 (  4,   89,   -1, 0)
89 STN  0 NEXT INTARV: next-pk 5 Next time  88.3905 ms  11049 ia 115.162468
89 STN  0 NEXT PK ARV: next-pk 5 Next time 203.5529 ms  25445
89 STN  0 TRANSMIT : SA  0 DA 10 pk   4 channel state  1 stn state (prev I next T) attempts 1
```

```
89 STN  4 CRA: wait  2
89 STN  9 CRA: wait  7
89 STN  0 RV ACK   : SA  0 DA 10 channel state  1 stn state (prev T next I) tx-count 1 Queue (H -1,T -1, L  0) (NM,SAV, iST,TX):
90 STN  4 CRA: wait  1
90 STN  9 CRA: wait  6
91 STN  4 TRANSMIT : SA  4 DA 10 pk   0 channel state  1 stn state (prev R next T) attempts 2
91 STN  9 CRA: wait  5
91 STN  4 RV ACK   : SA  4 DA 10 channel state  1 stn state (prev T next I) tx-count 2 Queue (H -1,T -1, L  0) (NM,SAV, iST,TX):
```

Així tenim que donat el valor aleatori concret realment la retransmissió passa a wait +2 essent el 2 l'slot detecció i el seguent el de començar el compte enrere. I aquest valor és sempre així. Per exemple l'estació 9 anterior li ha donat un wait de 9 slots per tant haurà de retransmetre a 85+9+2=96, i efectivament en la continuació de la traça anterior arribem a l'slot 96 on l'estació 9 retransmet:

```
92 STN  9 CRA: wait  4
93 STN  9 CRA: wait  3
94 STN  6 PK  ARRIVAL: pk (num   1, arv 93.739592 sarv   94 Tx-count  0) QU Queue (H  0,T  0, L  1) (NM,SAV, iST,TX):   0 (  1,   94,   -1, 0)
94 STN  6 NEXT INTARV: next-pk 2 Next time  93.7396 ms  11718 ia 52.019928
94 STN  6 NEXT PK ARV: next-pk 2 Next time 145.7595 ms  18220
94 STN  6 TRANSMIT : SA  6 DA 10 pk   1 channel state  1 stn state (prev I next T) attempts 1
94 STN  9 CRA: wait  2
94 STN  6 RV ACK   : SA  6 DA 10 channel state  1 stn state (prev T next I) tx-count 1 Queue (H -1,T -1, L  0) (NM,SAV, iST,TX):
95 STN  5 PK  ARRIVAL: pk (num   1, arv 94.212749 sarv   95 Tx-count  0) QU Queue (H  0,T  0, L  1) (NM,SAV, iST,TX):   0 (  1,   95,   -1, 0)
95 STN  5 NEXT INTARV: next-pk 2 Next time  94.2127 ms  11777 ia 6.712715
95 STN  5 NEXT PK ARV: next-pk 2 Next time 100.9255 ms  12616
95 STN  8 PK  ARRIVAL: pk (num   2, arv 94.322220 sarv   95 Tx-count  0) QU Queue (H  0,T  0, L  1) (NM,SAV, iST,TX):   0 (  2,   95,   -1, 0)
95 STN  8 NEXT INTARV: next-pk 3 Next time  94.3222 ms  11791 ia 26.004185
95 STN  8 NEXT PK ARV: next-pk 3 Next time 120.3264 ms  15041
95 STN  5 TRANSMIT : SA  5 DA 10 pk   1 channel state  1 stn state (prev I next T) attempts 1
95 STN  8 TRANSMIT : SA  8 DA 10 pk   2 channel state  2 stn state (prev I next T) attempts 1
95 STN  9 CRA: wait  1
96 STN  5 CRA DETERMINISTIC: wait  5
96 STN  5 COLLISION: SA -1 DA -1 pk   1 channel state  0 stn state (prev T next R) attempts 1 wait  5
96 STN  8 CRA DETERMINISTIC: wait  8
96 STN  8 COLLISION: SA -1 DA -1 pk   2 channel state  0 stn state (prev T next R) attempts 1 wait  8
96 STN  9 TRANSMIT : SA  9 DA 10 pk   3 channel state  1 stn state (prev R next T) attempts 2
96 STN  9 RV ACK   : SA  9 DA 10 channel state  1 stn state (prev T next I) tx-count 2 Queue (H -1,T -1, L  0) (NM,SAV, iST,TX):
```

6.a) 10.- Can we say with this trace that our model of the slotted aloha protocol is well implemented? Justify your explanation relation the answer to previous questions and adding any additional detailed comments not included in previous questions.

Answer: Això vol dir assegurar el bon funcionament d'un paquet: arribada-transmissió-possibles_transmissions-Resolució de col·lisions-ACKs-transmissió_seguent, a més de comprovar la seqüència d'arribades dels diferents paquets, les cues, i les sortides dels paquets.

Donat tots els detalls anteriors, hem comprovat tot menys els ACKs. No hem visualitzat els paquets ACK que indiquen a l'estació que realment la transmissió ha tingut èxit. Nosaltres ho podem inferir al mirar l'estat del canal a 1 però l'estació no sap l'estat del canal i realment es regeix amb el ACK. Com el procés evoluciona correctament podríem dir que l'ACk està ben implementat, però ho veiem en la següent configuració.

En aquest punt és important remarcar el primer paquet arriba a l'slot 1 i es transmet a l'slot 1 per tant pot tenir un retard de 0 que és físicament impossible. A la següent traça veiem perquè es dona això.

6.b) Configuration DEBUGCRA = 1 (and all other DEBUG to 0) with the in-ref file. Explain the differences of this trace with the previous one and explain what use this one has and what functionality it has lost from the previous trace. Do so explaining in detailed exact lines of traces.

Answer: Ara veiem la seqüència neta Arribada-Transmissió-ACK de cada paquet sense cap altre tipus de missatges. Veiem doncs que el primer paquet rep l'ACK al mateix moment que hi ha l'arribada.

```
1 STN  1 PK  ARRIVAL: pk (num   0, arv  0.239988 sarv    1 Tx-count  0) QU Queue (H  0,T  0, L  1) (NM,SAV, iST,TX):   0 (  0,    1,   -1, 0)
1 STN  1 TRANSMIT : SA  1 DA 10 pk   0 channel state  1 stn state (prev I next T) attempts 1
1 STN  1 RV ACK   : SA  1 DA 10 channel state  1 stn state (prev T next I) tx-count 1 Queue (H -1,T -1, L  0) (NM,SAV, iST,TX):
```

És a dir al temps mínim hi falta els temps de transmissió del paquet i de l'ACK. Hauríem d'afegir una constant de 2 slots a les estadístiques de retard. Com aquest codi al final el farem servir per comparar algorismes de CRA, afegir o no aquesta constant és irrellevant ja que el que tardi menys continuarà tardant menys, i vice-versa.

A continuació veiem una traça d'una collisió a l'slot 13 entre les estacions 5 i 8 per la transmissió del paquet 0 i 2 respectivament:

```
13 STN  5 TRANSMIT : SA  5 DA 10 pk   0 channel state  1 stn state (prev I next T) attempts 1
13 STN  8 TRANSMIT : SA  8 DA 10 pk   2 channel state  2 stn state (prev I next T) attempts 1
14 STN  5 CRA DETERMINISTIC: wait  5
14 STN  5 COLLISION: SA -1 DA -1 pk   0 channel state  0 stn state (prev T next R) attempts 1 wait  5
14 STN  8 CRA DETERMINISTIC: wait  8
14 STN  8 COLLISION: SA -1 DA -1 pk   2 channel state  0 stn state (prev T next R) attempts 1 wait  8
15 STN  5 CRA: wait  5
15 STN  8 CRA: wait  8
16 STN  2 PK  ARRIVAL: pk (num   1, arv 15.520403 sarv   16 Tx-count  0) QU Queue (H  0,T  0, L  1) (NM,SAV, iST,TX):   0 (  1,   16,   -1, 0)
16 STN  2 TRANSMIT : SA  2 DA 10 pk   1 channel state  1 stn state (prev I next T) attempts 1
16 STN  5 CRA: wait  4
16 STN  8 CRA: wait  7
```

La col·lisió es detecta a l'slot 14 i comencen les dues estacions a esperar 5 i 8 slots respectivament. La traça visualitza el compte enrere del temps d'espera i es veu que les dues resten un el temps d'espera en cada slot. El seguent intent de retransmissió de l'estació 5 ha de ser a l'slot 20 (14+5+1) i de l'estació 8 a l'slot 23 (14+8+1).

Veiem les transmissions dels paquets 0 i 2 de les estacions 5 i 8

```
20 STN  5 TRANSMIT : SA  5 DA 10 pk   0 channel state  3 stn state (prev R next T) attempts 2
```

```
23 STN  8 TRANSMIT : SA  8 DA 10 pk   2 channel state  1 stn state (prev R next T) attempts 2
```

Al perdre el càlcul d'inter-arrival i next arrival, ja no podem fer el seguiment detallat del generador de paquets. No podem saber si el següent paquet que es genera es genera quan toca.

6.c) Configuration DEBUGchannel = 1(and all other DEBUG to 0), with the in-ref file. Explain the use of this trace to follow the protocol operations and provide concrete examples of what it shows and why. Explain the differences of this trace and the previous traces and what functionalities it provides that the other do not have and which ones do not have that the previous ones have.

Answer: Aquesta traça és molt diferent. Aquí visualitzem l'evolució del canal, per tant la traça ara observa només el què passa pel canal. Per això cal visualitzar el què té el canal en cada slot: 1) al començar l'slot, 2) visualitzar operacions que les estacions facin en aquest slot sobre el canal 3) i consultar el canal al final de l'slot per interpretar què ha passat en aquest slot. El canal transmet paquets per això es veu adreça origen (SA) adreça destí (DA) numero paquet en transmissió (només vàlid si no hi ha col·lisió), estat de l'slot indicant quantes transmissions porta acumulades aquest slot, i alguns paràmetres més ja vistos en missatges anteriors.

Així tenim que la primera línia d'un slot és sempre:

```
0 Creating new Slot empty...channel SA -1 DA -1 S  0 pk ( -1,  -1.0000,   -1, -1)
```

Això posa l'slot a buit i preparat perquè estacions comencin a transmetre. Els valors representen l'estat de repòs del canal sense transmissió, i és el valor inicial en tots i cada un dels slots simulats.

Així tenim que la última línia d'un slot és sempre:

```
0 END SLOT TIME:.......... channel SA....
```

Indica situació final de l'slot que ara sí pot tenir diferents valors en els paràmetres segons les operacions sobre el canal en aquest slot.

Al final de l'slot l'estació especial que fa de destí, mira la informació en l'slot i determina si hi ha hagut col·lisió (`CHANNEL COLLISION: SA ....`) i sinó envia l'ACK a la única estació que ha transmès en aquest slot. En les traces anteriors el missatge de col·lisió no es visualitza però ho fa igualment. Igual que ara genera també l'ACK però no ho visualitza perquè no estem verificant l'operació de l'estació sinó del canal.

Aquí veiem per tant el següent tros de traça corresponent a l'slot 133 com exemple:

```
132 END SLOT TIME:.......... channel SA -1 DA -1 S  0 pk ( -1,   -1,   -1, -1)
133 Creating new Slot empty...channel SA -1 DA -1 S  0 pk ( -1,  -1.0000,   -1, -1)
133 STN  2 TRANSMIT : SA  2 DA 10 pk   2 channel state  1 stn state (prev I next T) attempts 1
133 STN  4 TRANSMIT : SA  4 DA 10 pk   7 channel state  2 stn state (prev I next T) attempts 1
133 STN  6 TRANSMIT : SA  6 DA 10 pk   3 channel state  3 stn state (prev R next T) attempts 2
133 END SLOT TIME:.......... channel SA  6 DA 10 S  3 pk (  3,  125,  125,  1)
133 CHANNEL COLLISION: SA  6 DA 10 pk   3 channel state  3
134 Creating new Slot empty...channel SA -1 DA -1 S  0 pk ( -1,  -1.0000,   -1, -1)
```

Veiem que hi ha 3 estacions transmetent i sobreescrivint informació al canal en el mateix slot. Al acabar l'slot l'estat és 3, i per tant es considera col·lisió. I es passa al següent slot.
A continuació veiem l'slot 135 que no passa cap operació sobre el canal, i per tant l'estat del canal no varia. I l'slot 136 que només transmet l'estació 7 i per tant és una transmissió correcte i no apareix el missatge de col·lisió.

```
135 END SLOT TIME:.......... channel SA -1 DA -1 S  0 pk ( -1,   -1,   -1, -1)
136 Creating new Slot empty...channel SA -1 DA -1 S  0 pk ( -1,  -1.0000,   -1, -1)
136 STN  7 TRANSMIT : SA  7 DA 10 pk   4 channel state  1 stn state (prev R next T) attempts 3
136 END SLOT TIME:.......... channel SA  7 DA 10 S  1 pk (  4,  118,  118,  2)
137 Creating new Slot empty...channel SA -1 DA -1 S  0 pk ( -1,  -1.0000,   -1, -1)
```

6.d) What DEBUG configuration would be needed to verify all protocol functionalities (but without including more messages than needed)? Include an example of this proposed trace and explain the full protocol operation over this trace.

Answer: Per verificar-ho tot a l'hora caldria activar tots els missatges a l'hora i seria una traça molt llarga i carregosa de seguir i verificar. No es dupliquen missatges perquè si un missatge es necessita per més d'una funcionalitat només apareix un cop. Aquí tenim un tall del què passa els 10 primers slots:

```
  0 Creating new Slot empty...channel SA -1 DA -1 S  0 pk ( -1,  -1.0000,  -1, -1)
  0 END SLOT TIME:.......... channel SA -1 DA -1 S  0 pk ( -1,  -1,  -1, -1)
  1 Creating new Slot empty...channel SA -1 DA -1 S  0 pk ( -1,  -1.0000,  -1, -1)
  1 CREATING NEW ELEMENT: num 0 atime  0.2400 stime    1 tx-count 0
  1 ADD QUEUE BEFORE: pos -1 elem (num   0, arv  0.2400 sarv    1 txcnt  0): Queue (H -1,T -1, L  0) (NM,SAV, iST,TX):
  1 ADD QUEUE... pos 0 elem (  0,  0.2400    1  0): Queue (H  0,T  0, L  1) (NM,SAV, iST,TX):  0 (  0,   1,  -1, 0)
  1 STN  1 PK  ARRIVAL: pk (num   0, arv  0.239988 sarv   1 Tx-count  0) QU Queue (H  0,T  0, L  1) (NM,SAV, iST,TX):  0 (  0,   1,  -1, 0)
  1 STN  1 NEXT INTARV: next-pk 1 Next time  0.2400 ms    30 ia 11.587130
  1 STN  1 NEXT PK ARV: next-pk 1 Next time  11.8271 ms   1479
  1 STN  1 TRANSMIT : SA  1 DA 10 pk   0 channel state  1 stn state (prev I next T) attempts 1
  1 END SLOT TIME:.......... channel SA  1 DA 10 S  1 pk (  0,   1,  -1,  0)
  1 DELETE QUEUE BEFORE ... pos 0 elem (  0,  0.2400,    1) Queue (H  0,T  0, L  1) (NM,SAV, iST,TX):  0 (  0,   1,   1, 1)
  1 DELETE QUEUE AFTER  ... pos 0   1 CREATING NEW ELEMENT: num -1 atime  -1.0000 stime    -1 tx-count 0
elem (  0,  0.2400,    1) Queue (H -1,T -1, L  0) (NM,SAV, iST,TX):
  1 STN  1 RV ACK   : SA  1 DA 10 channel state  1 stn state (prev T next I) tx-count 1 Queue (H -1,T -1, L  0) (NM,SAV, iST,TX):
  2 Creating new Slot empty...channel SA -1 DA -1 S  0 pk ( -1,  -1.0000,  -1, -1)
  2 END SLOT TIME:.......... channel SA -1 DA -1 S  0 pk ( -1,  -1,  -1, -1)
  3 Creating new Slot empty...channel SA -1 DA -1 S  0 pk ( -1,  -1.0000,  -1, -1)
  3 CREATING NEW ELEMENT: num 0 atime  2.1511 stime    3 tx-count 0
  3 ADD QUEUE BEFORE: pos -1 elem (num   0, arv  2.1511 sarv    3 txcnt  0): Queue (H -1,T -1, L  0) (NM,SAV, iST,TX):
  3 ADD QUEUE... pos 0 elem (  0,  2.1511    3  0): Queue (H  0,T  0, L  1) (NM,SAV, iST,TX):  0 (  0,   3,  -1, 0)
  3 STN  4 PK  ARRIVAL: pk (num   0, arv  2.151097 sarv   3 Tx-count  0) QU Queue (H  0,T  0, L  1) (NM,SAV, iST,TX):  0 (  0,   3,  -1, 0)
  3 STN  4 NEXT INTARV: next-pk 1 Next time  2.1511 ms    269 ia 38.592656
  3 STN  4 NEXT PK ARV: next-pk 1 Next time  40.7438 ms   5093
  3 STN  4 TRANSMIT : SA  4 DA 10 pk   0 channel state  1 stn state (prev I next T) attempts 1
  3 END SLOT TIME:.......... channel SA  4 DA 10 S  1 pk (  0,   3,  -1,  0)
  3 DELETE QUEUE BEFORE ... pos 0 elem (  0,  2.1511,    3) Queue (H  0,T  0, L  1) (NM,SAV, iST,TX):  0 (  0,   3,   3, 1)
  3 DELETE QUEUE AFTER  ... pos 0   3 CREATING NEW ELEMENT: num -1 atime  -1.0000 stime    -1 tx-count 0
elem (  0,  2.1511,    3) Queue (H -1,T -1, L  0) (NM,SAV, iST,TX):
  3 STN  4 RV ACK   : SA  4 DA 10 channel state  1 stn state (prev T next I) tx-count 1 Queue (H -1,T -1, L  0) (NM,SAV, iST,TX):
  4 Creating new Slot empty...channel SA -1 DA -1 S  0 pk ( -1,  -1.0000,  -1, -1)
  4 CREATING NEW ELEMENT: num 0 atime  3.9906 stime    4 tx-count 0
  4 ADD QUEUE BEFORE: pos -1 elem (num   0, arv  3.9906 sarv    4 txcnt  0): Queue (H -1,T -1, L  0) (NM,SAV, iST,TX):
  4 ADD QUEUE... pos 0 elem (  0,  3.9906    4  0): Queue (H  0,T  0, L  1) (NM,SAV, iST,TX):  0 (  0,   4,  -1, 0)
  4 STN  2 PK  ARRIVAL: pk (num   0, arv  3.990618 sarv   4 Tx-count  0) QU Queue (H  0,T  0, L  1) (NM,SAV, iST,TX):  0 (  0,   4,  -1, 0)
  4 STN  2 NEXT INTARV: next-pk 1 Next time  3.9906 ms    499 ia 11.529785
  4 STN  2 NEXT PK ARV: next-pk 1 Next time  15.5204 ms   1941
  4 STN  2 TRANSMIT : SA  2 DA 10 pk   0 channel state  1 stn state (prev I next T) attempts 1
  4 END SLOT TIME:.......... channel SA  2 DA 10 S  1 pk (  0,   4,  -1,  0)
  4 DELETE QUEUE BEFORE ... pos 0 elem (  0,  3.9906,    4) Queue (H  0,T  0, L  1) (NM,SAV, iST,TX):  0 (  0,   4,   4, 1)
  4 DELETE QUEUE AFTER  ... pos 0   4 CREATING NEW ELEMENT: num -1 atime  -1.0000 stime    -1 tx-count 0
elem (  0,  3.9906,    4) Queue (H -1,T -1, L  0) (NM,SAV, iST,TX):
  4 STN  2 RV ACK   : SA  2 DA 10 channel state  1 stn state (prev T next I) tx-count 1 Queue (H -1,T -1, L  0) (NM,SAV, iST,TX):
  5 Creating new Slot empty...channel SA -1 DA -1 S  0 pk ( -1,  -1.0000,  -1, -1)
  5 END SLOT TIME:.......... channel SA -1 DA -1 S  0 pk ( -1,  -1,  -1, -1)
  6 Creating new Slot empty...channel SA -1 DA -1 S  0 pk ( -1,  -1.0000,  -1, -1)
  6 CREATING NEW ELEMENT: num 0 atime  5.5361 stime    6 tx-count 0
  6 ADD QUEUE BEFORE: pos -1 elem (num   0, arv  5.5361 sarv    6 txcnt  0): Queue (H -1,T -1, L  0) (NM,SAV, iST,TX):
  6 ADD QUEUE... pos 0 elem (  0,  5.5361    6  0): Queue (H  0,T  0, L  1) (NM,SAV, iST,TX):  0 (  0,   6,  -1, 0)
  6 STN  8 PK  ARRIVAL: pk (num   0, arv  5.536081 sarv   6 Tx-count  0) QU Queue (H  0,T  0, L  1) (NM,SAV, iST,TX):  0 (  0,   6,  -1, 0)
  6 STN  8 NEXT INTARV: next-pk 1 Next time  5.5361 ms    693 ia 1.383660
  6 STN  8 NEXT PK ARV: next-pk 1 Next time  6.9197 ms    865
  6 STN  8 TRANSMIT : SA  8 DA 10 pk   0 channel state  1 stn state (prev I next T) attempts 1
  6 END SLOT TIME:.......... channel SA  8 DA 10 S  1 pk (  0,   6,  -1,  0)
  6 DELETE QUEUE BEFORE ... pos 0 elem (  0,  5.5361,    6) Queue (H  0,T  0, L  1) (NM,SAV, iST,TX):  0 (  0,   6,   6, 1)
  6 DELETE QUEUE AFTER  ... pos 0   6 CREATING NEW ELEMENT: num -1 atime  -1.0000 stime    -1 tx-count 0
elem (  0,  5.5361,    6) Queue (H -1,T -1, L  0) (NM,SAV, iST,TX):
```

```
  6 STN  8 RV ACK   : SA  8 DA 10 channel state  1 stn state (prev T next I) tx-count 1 Queue (H -1,T -1, L  0) (NM,SAV, iST,TX):
  7 Creating new Slot empty...channel SA -1 DA -1 S  0 pk ( -1,  -1.0000,   -1, -1)
  7 CREATING NEW ELEMENT: num 1 atime   6.9197 stime      7 tx-count 0
  7 ADD QUEUE BEFORE: pos -1 elem (num    1, arv   6.9197 sarv      7 txcnt  0): Queue (H -1,T -1, L  0) (NM,SAV, iST,TX):
  7 ADD QUEUE... pos 0 elem (   1,   6.9197      7  0): Queue (H  0,T  0, L  1) (NM,SAV, iST,TX):   0 (  1,    7,   -1, 0)
  7 STN  8 PK  ARRIVAL: pk (num   1, arv  6.919741 sarv    7 Tx-count  0) QU Queue (H  0,T  0, L  1) (NM,SAV, iST,TX):   0 (  1,    7,   -1, 0)
  7 STN  8 NEXT INTARV: next-pk 2 Next time   6.9197 ms    865 ia 5.652291
  7 STN  8 NEXT PK ARV: next-pk 2 Next time  12.5720 ms   1572
  7 STN  8 TRANSMIT : SA  8 DA 10 pk   1 channel state  1 stn state (prev I next T) attempts 1
  7 END SLOT TIME:..... channel SA  8 DA 10 S  1 pk (  1,    7,   -1,  0)
  7 DELETE QUEUE BEFORE ... pos 0 elem (   1,   6.9197,      7) Queue (H  0,T  0, L  1) (NM,SAV, iST,TX):   0 (  1,    7,    7, 1)
  7 DELETE QUEUE AFTER  ... pos 0    7 CREATING NEW ELEMENT: num -1 atime  -1.0000 stime     -1 tx-count 0
elem (   1,   6.9197,      7) Queue (H -1,T -1, L  0) (NM,SAV, iST,TX):
  7 STN  8 RV ACK   : SA  8 DA 10 channel state  1 stn state (prev T next I) tx-count 1 Queue (H -1,T -1, L  0) (NM,SAV, iST,TX):
  8 Creating new Slot empty...channel SA -1 DA -1 S  0 pk ( -1,  -1.0000,   -1, -1)
  8 END SLOT TIME:..... channel SA -1 DA -1 S  0 pk ( -1,   -1,   -1, -1)
  9 Creating new Slot empty...channel SA -1 DA -1 S  0 pk ( -1,  -1.0000,   -1, -1)
  9 END SLOT TIME:..... channel SA -1 DA -1 S  0 pk ( -1,   -1,   -1, -1)
 10 Creating new Slot empty...channel SA -1 DA -1 S  0 pk ( -1,  -1.0000,   -1, -1)
 10 CREATING NEW ELEMENT: num 0 atime   9.9912 stime     10 tx-count 0
 10 ADD QUEUE BEFORE: pos -1 elem (num    0, arv   9.9912 sarv     10 txcnt  0): Queue (H -1,T -1, L  0) (NM,SAV, iST,TX):
 10 ADD QUEUE... pos 0 elem (   0,   9.9912     10  0): Queue (H  0,T  0, L  1) (NM,SAV, iST,TX):   0 (  0,   10,   -1, 0)
 10 STN  0 PK  ARRIVAL: pk (num   0, arv  9.991196 sarv   10 Tx-count  0) QU Queue (H  0,T  0, L  1) (NM,SAV, iST,TX):   0 (  0,   10,   -1, 0)
 10 STN  0 NEXT INTARV: next-pk 1 Next time   9.9912 ms   1249 ia 9.447202
 10 STN  0 NEXT PK ARV: next-pk 1 Next time  19.4384 ms   2430
 10 STN  0 TRANSMIT : SA  0 DA 10 pk   0 channel state  1 stn state (prev I next T) attempts 1
 10 END SLOT TIME:..... channel SA  0 DA 10 S  1 pk (  0,   10,   -1,  0)
 10 DELETE QUEUE BEFORE ... pos 0 elem (   0,   9.9912,     10) Queue (H  0,T  0, L  1) (NM,SAV, iST,TX):   0 (  0,   10,   10, 1)
 10 DELETE QUEUE AFTER  ... pos 0   10 CREATING NEW ELEMENT: num -1 atime  -1.0000 stime     -1 tx-count 0
elem (   0,   9.9912,     10) Queue (H -1,T -1, L  0) (NM,SAV, iST,TX):
 10 STN  0 RV ACK   : SA  0 DA 10 channel state  1 stn state (prev T next I) tx-count 1 Queue (H -1,T -1, L  0) (NM,SAV, iST,TX):
```

Una traça així tan detallada és massa per verificar tot de cop. És important poder verificar les parts independents per separat, i si són independents no cal verificar la traça conjunta, ja que qualsevol traça té darrera l'execució conjunta (és sempre la mateixa execució complerta del protocol) i només tenim visualització parcial dels missatges activant o no la impressió segons ens interessi.

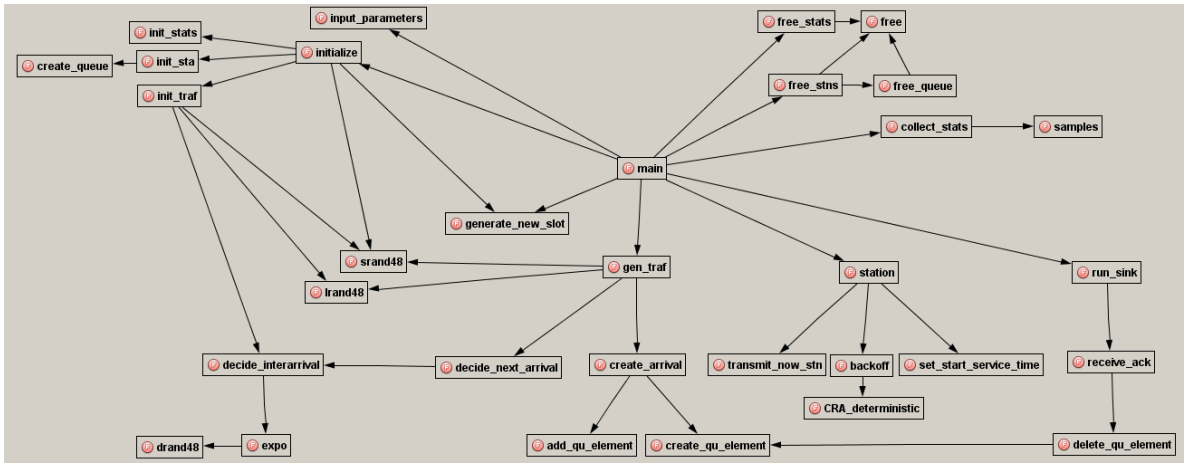
(Optional pre-submission must have done up to here at least, or more. Try to at least review the handout up to the end to try to identify your doubts to finish it so we can discuss them in the practice session)

PART III: Code Understanding

Up to here we have analyzed the program from outside (the output) with the traces and results, and we have understood how the protocol works. Make sure you fully understand the protocol operation of the previous section, otherwise it will be very difficult to try to understand the protocol directly from the code. Now we want to understand the **structure of the code** that implements it. Get into the code and answer the following questions about its design and implementation. Remember that the understanding is important as this code will be used for most of rest of the practice work of the course. On the other hand, we'll go slowly and we'll take the entire course to go through all the elements that build the simulation. Hence, we do not expect to understand now all the details of each single function in the simulation. Instead, we expect an understanding at the level of program structure. The detail of many functions will be worked out in the following theory chapters and practices. Hence it is important to focus now in what it is requested now and not to try to go beyond as we have not done the theory yet to understand some of the concepts already included in the simulation.

The program models the network defining the communication cannel (the cannel) and the users that transmit over the network as stations (stns).

The program structure is shown in the following diagram (obtained with the call graph netbeans function while editing the code with netbeans):



Question 7: CHANNEL – The program models the cannel adding the transmission signal (information) of each transmitting station in each (time) slot executed. The program executes the operations that each station must do at the current slot and it updates the statistics before advancing to the next time slot. This way, the channel can be modeled with just one slot (there is no need of a vector of slots as only one is active at each point in time). The slot must be initialized each time the simulation advances to the next time slot (at each iteration of the simulation).

7.a) Analyze the sslot and schannel data structures to understand how the channel is modeled inside the simulation.

7.a) 1.- Understand and explain in your own words the fields included in the structures sslot and sschanel and explain the relation between these two structures and why it makes sense to define two (instead of just one).

Answer: l'estructura sslot inclou la informació que es transmet en el canal. Com no utilitzem els paquets que es transmeten realment només s'ulititza el **SA** per l'estació destí saber qui ha enviat el paquet. El DA és sempre el mateix i podríem no haver-lo inclòs en el model perquè no es fa servir. El state indica l'estat de la transmissió que això és el model que representa el senyal en el canal. Per això serveix per identificar si es transmet (>0) o no (=0) i si es transmet si hi ha col·lisió (>1) o no (=1). El camp pk és tota la informació del paquet que s'està transmeten (que serà vàlid només quan no hi ha col·lisió) l'estructura equeue el paquet està definida amb la informació necessària del paquet dins la simulació, i no amb els camps del paquet real que es transmetria. En realitat el paquet en sí només es fa servir per les traces i verificacions.

Si ens fixem la informació a sslot és la informació "física" de senyal que enviaríem pel canal.

L'estructura schannel són paràmetres que caracteritzen físicament i funcionament el canal: Rate és la velocitat de transmissió del canal (10Mbps, 100Mbps, 1Gbps...); stot_size identifica el tamany en què dividim el canal per les transmissions (que és el temps de transmissió dels bytes de l'slot_size); Cralg indica l'algorisme de resolució de col·lisions que apliquem; p és el paràmetre per un CRA determinat; I cslot és realment la informació que es transmet en cada slot del tipus sslot tal i com s'ha explicat.

Així doncs el schannel és el canal amb tota la definició i informació. I el sslot només és la informació que es transmetria pel canal.

7.a) 2.- Why the state field in sslot is defined as an int and what does this number indicate at any time?

Answer: Perquè indica l'estat de la senyal en el canal en cada moment: 0 indica no hi ha senyal; 1 indica hi ha una transmissió i per tant el senyal és intel·ligible; >1 indica vàries transmissions a l'hora i per tant col·lisió inintel·ligible.

7.a) 3.- Why sslot has a packet in it (pk field)?

Answer: Seria la informació del paquet que s'està transmetent. Però notar que la informació que hi posem en la simulació no és la mateixa que posaríem realment en el canal (els camps del paquet serien: DA, SA, ...).

7.b) Explain in your own words what the function generate_new_slot does. Why the state field of the slot is set to EMPTY in this function?

Answer: This function prepares the channel for the next slot, hence it gets ready to move the simulation time to the following

slot and the following slot has to be empty as it has not occurred yet in the simulation time. The actions during the slot will fill this slot with information (or garbage in case of collision) or leave it empty if no transmission from any station occurs in this slot. The EMPTY state has the 0 value which corresponds to no transmission (yet) in this slot.

Note that all operation happens on a single slot. Hence the simulation can only have the current slot as state and it does not need to keep track of several slots at the same time. We would need several slots in an scenario where the transmission time takes several slots and hence from the starting slot to the end slot of transmission other actions may happen in other processes of the simulation.

Question 8: STATIONS – The stations generate packet, store them in the queue and transmit them to the channel.

8.a) Analyze the **sstation** structure which includes the information that models the station which in turn is defined using the **squeue** and **equeue** structures. Study these structures to understand the following concepts in the implementation:

8.a) 1.- Explain the structure definition explaining each field in them and justifying the type used to define each field.

Answer: l'estructura equeue conté la informació d'un paquet en concret inclou: el seu número únic dins l'estació, temps d'arribada (en decimal i l'slot que toca), temps de servei, i vegades que s'ha intentat transmetre.

L'estructura squeue és la cua que s'implementa com una cua circular que té una longitud (lng), primer de la cua (head) d'on es treuen elements, final de la cua (tail) on s'afegeixen els paquets, i max és la capacitat màxima de paquets que pot tenir la cua. Els paquets estan guardats a pks que és un vector on s'implementa la cua circular. Cada paquet és del tipus equeue definit abans.

L'estructura sstation defineix l'estació amb número d'estació únic a la xarxa (stnnum), velocitat d'arribada de paquets (rate), quin és el temps de la següent arribada per aquesta estació (nextpkarv), l'estat per fer el seguiment del protocol (I/R/T) (state), la cua de paquets d'estructura la indicada abans (qu), el nombre total de paquets creats (tpk), la probabilitat p per aplicar al p-persistence (no fet servir ara), temps de transmissió del paquet en curs de l'estació (txtslot), nombre d'eslots a esperar abans de retransmetre (wait).

8.a) 2.- How is the packet modeled? What structure defines the packet and types are used to define it and its fields? Where are the packets stored?

Answer: el concepte de paquet el defineix el equeue que ja hem explicat el seu contingut. I els paquets es guarden a la cua de les estacions.

8.a) 3.- How is the queue of packet being modeled? What type of queue is used to implement it (static or dynamic, linked, single-link or double-linked, circular...) How many packets can the queue store (indicate the constant that defines it)? How many packet queues are defined in the simulation? Where are they stored? Justify the answers.

Answer: la cua de paquets es modela amb l'estructura squeue ja explicada i que implementa una cua circular perquè quan s'afegeix a la cua (add_qu_element) si s'ha arribat al MAXQU es torna a començar a l'index (tail) zero si hi ha posicions buides per reutilitzar. Dins la cua hi ha el punter als paquets però no està definit quants. Si mirem la creació de la cua veurem que és d'un màxim MAXQU.

Hi ha una cua per cada estació, per tant hi ha nstns cues a la simulació. Aquestes cues es guarden dins l'estació.

8.b) The sloha protocol operation is described with an state diagram of 3 states [2]. Each state represents a diferent scenario for transmission in the station. Understand this operation answering the following questions:

8.b) 1.- Name each of the states the station can be in. Explain what transmission conditions defined each state. (Hint: check the stations state #define in the code)

Answer: 3 estats: Idle (buit) que no està fent res, T (transmitting) que ha enviat un paquet i espera el ack, R (resolució) que està esperant els slots requerits abans de poder retransmetre.

8.b) 2.- How is the time to wait before retransmission after a collision (CRA) determined? What functions are involved and how are they related?

Answer: es calcula a la funció backoff que se li passa el número de l'estació que demana que se li calcula el temps d'espera.

Es pot aplicar varis algorismes pel càlcul. Ara hi ha el deterministic que retorna el mateix valor per cada estació, però les diferents estacions tenen un valor diferent perquè no torni a col·lisionar. El valor que retorna és el mateix número d'estació. L'estació 0 tindrà clarament preferència perquè sempre esperarà 0. En contra partida l'estació amb número més alt tindrà una clara desavantatge perquè esperarà sempre més temps (slots).

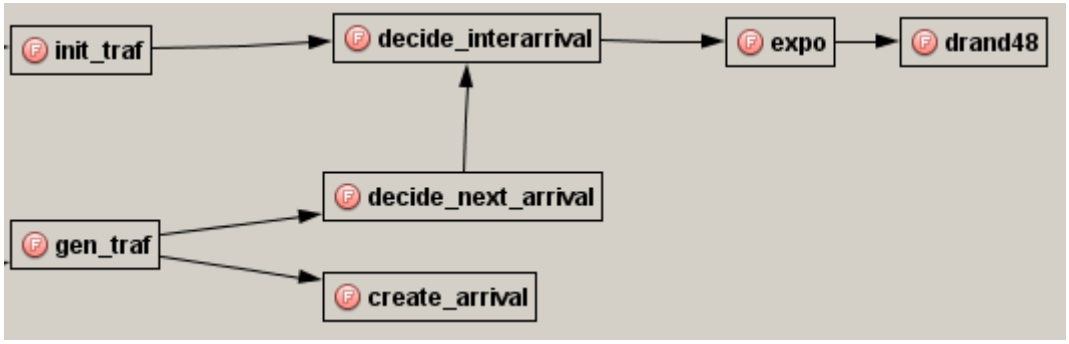
Question 9: TRAFIC GENERATOR – The packet generator is implemented with the functions init_traf() and gen_traf(). It computes the slot time that needs to be generated the next packet at each station (decide_next arrival), and when the simulation reaches this time it creates the packet (create_arrival). The arrival time is defined is defined as an incremental time (hence it is an inter-arrival value) from the time of current arrival (decide_interarrival). At this point, neglect all code related to randomness (rand, seed) as it is studied in another practice.

9.a) Explain what each of the mentioned functions do (the how it is done it will be analyzed in another practice) and explain how these functions relate to each other and do a block diagram to visualize this relations.

Answer: La funció decide_interarrival retorna un temps entre arribades seguint una funció de distribució definida. init-traf és la funció que determina l'slot que es generarà el primer paquet. Per tant crida decide-interarrival per decidir el temps entre inici de simulació i primer paquet.

El programa principal crida en cada slot el gen_traf que és el generador de paquets de totes les estacions. Cada estació té el temps de la següent arribada (nextpkarv). El gentraf consulta per cada estació si ha de generar un paquet a l'slot actual i si és el cas el genera amb la funció create_arrival, i llavors decideix quan s'ha de generar el següent paquet per aquesta estació (decide_next_arrival).

A continuació tenim el diagrama de blocs que es veu que el gen_traf decideix els next arrivals i crea paquets. Per decidir el next arrival es fa servir el decide_interarrival. I el temps de la primera arrivada (nextpkarv) es decideix a init-traf.



9.b) Why can a packet be transmitted at the same slot it arrives?

Answer: Perquè en realitat el paquet arriba en algun moment de l'slot anterior, i per tant el paquet està disponible a l'inici de l'slot en que es comptabilitza que arriba. Es guarden els dos temps d'arribada (discret i continu) per si es vol calcular el temps de retard en continu. En termes d'implementació, la generació del paquet està abans de la transmissió per tant el flux d'execució del codi fa que la transmissió vingui després i per tant es hi ha l'opció de transmetre el paquet dins la mateixa iteració del bucle slot (al main). I per tant això vol dir que s'executa en el mateix slot.

9.b) 1.- What does the simulation do with the messages/packets successfully transmitted in the channel? Who receives them? What happen to them after it? Justify the answers.

Answer: Els processa el sink per decidir activar l'ack o no. I llavors s'elimina al resetejar el canal amb el generate_new_slot.

Question 10: SINK – There is a special station called sink that takes the reception role for all packets sent in the network. Hence, it is the responsible station to send the ack to the sender of the successfully transmitted packets. Understand the functionality of this station answering the following questions

10.a) When is the sink station run (run_sink) and why?

Answer: S'executa al final de cada slot i per cada slot en el main. Això és degut a què ha d'actuar després de que totes les estacions hagin transmès per poder mirar el canal i rebre el missatge per generar el ack (o la col·lisió).

10.b) Explain how the process of sending and receiving the ack is implemented in the simulation, this is, i) How does the sink send the ack and ii) how the sending station receives it to consider the original data packet has been successfully transmitted.

Answer:

The ack is not transmitted to the channel but instead the sink station, who has received the packet, activates the process to modify the state of the sending station as it knows the packet it has been received successfully. Hence, the simulation violates the real world, as one station is physically different from another, and something known in one needs to be sent through the channel to be known in another station. However, the simulation can just execute the receive_ack function with parameter the sending station in the sink function. Note the function is called received_ack instead of transmit_ack, as there is no new packet of type ack sent in the channel (which could suffer collision too, but this is not contemplated/implemented in the simulation). The sink triggers the receive_ack which involves the elimination of the data packet in the queue, the change of the state (to idle) of the receiving station, and the update of statistics to count for the successful transmission of the packet being confirmed by the receiving ack.

10.c) If a packet has been successfully transmitted at slot s, at what slot does the ack arrive to the sending station? Justify your answer including a portion of a trace that helps explain the reasoning.

Answer: Com el sink s'executa al final del mateix slot es pot entregar l'ack a l'estació receptora al mateix slot que s'ha transmès. L'ack implica ja eliminar el paquet i guardar estadístiques per tant tenim un retard de transmissió i entrega de l'ack de 0 slots. Si mirem les traces es veu que un paquet pot ser creat, enviat i ack rebut tot al mateix slot. Però com això és físicament inviable les estadístiques incrementen aquest retard de 0 per 1 slot (almenys té que haver-hi 1 slot de retard que com a mínim seria el temps d'enviar el ACK, també en el mon real el sol processat de generació del paquet faria que no es pot transmetre al mateix slot). Així el càlcul de retard perquè no doni retards de zero es calcula: slot-pk.sarv_time+1 (al fer sts.dhist[stsstate][s->stnnum][slot-pk.sarv_time+1]++ dins la funció receive_ack). El més 1 és degut que slot pot ser el mateix que la generació.

10.d) What is the minimum time an ack packet would take to reach the sending station in the real world assuming the ack does not collide? Can the ack packet suffer collision in the real world? Justify your answers.

Answer: Doncs tindríem l'slot que rebem el paquet, i com a molt podríem enviar el paquet al següent slot. Si processar el paquet i generar el ack fos molt ràpid (instantani) l'ack es pot enviar exactament al següent slot. Si el temps de processat fos més no és podria enviar l'ack fins varis slots després.

El paquet ack en principi s'envia pel mateix canal i per tan podria tenir col·lisions, i per tant pot tardar molt o molt poc a arribar, és incert el temps que pot tardar. Però com a mínim al ser un paquet separat tardaria almenys un slot més.

Així el temps mínim que tardaria en rebre el ack seria 2 slots.

10.e) Is the ack transmission model a valid model?

Answer: Per segons quin tipus d'estudi es faci no seria vàlid. Per exemple per saber el temps total que es tarda a enviar un paquet caldria incloure el temps d'envio de l'ack. Però si només volem mesurar l'eficiència de l'algorisme de resolució de col·lisions n'hi ha prou. Ja que tots aquests temps addicionals són els mateixos per tots els algorismes i per tant no varia l'anàlisi de la comparativa.

Question 11: STATISTICS – Finally, the program keeps information of the variables values to get measures of the protocol behavior. It is important to take the correct data, at the correct point and at the precise time, to store them well and provide good results summary for being able to interpret them and derive conclusions from them. All of knowledge will be worked out along the quarter. Right now we want you to understand the data structure used to store the statistics and the computation of

the values computed in the basic version (as example). There are many fields already prepared for multiple statistics but we'll just focus on the output of the basic version as these fields are used to compute statistics in the full version.

11.a) What fields in the sstats structure are involved in the computation of the offered load and utilization? Explain what each of these two measures have to computer and how its computation is implemented.

Answer: offered load utilitza el gload que comptabilitza el nombre de paquets generats per cada estació. La utilització fa servir els paquets enviats snt. Els dos són long perquè guarden quants paquets d'una estació s'han generat o enviat. Una simulació pot durar molts slots per tant, per no tenir límits amb un int, el fem long. Els dos tenen dues dimensions. Una dimensió són estats d'estadístiques que veurem més endavant i l'altre dimensió és per estació. Per tant podem dir que és un vector que per cada estació té un número que és el nombre d'observacions de paquets generats o enviats respectivament.

11.b) Identify inside the code where these fields are updated and explain why the computation is correct.

Answer: el gload s'incrementa en 1 cada vegada que es genera un paquet per tant s'actualitza a la funció create_arrival.

Un paquet es considera enviat quan es rep a l'ack, en aquest punt es treu de la cua i s'elimina. Per això l'actualització del snt està a la funció receive_ack.

11.c) Explain the offered load and utilization formulas implemented in collect_stats function. Reason what variables use and why and explain why this obtains the target value of each of the metric.

Answer: les dues són taxes per tant quantitat per unitat de temps. Així el càlcul és la suma total de paquets generats o enviats dividit per la durada de la simulació que és nslots. D'aquests nslots se li resta start_stats perquè veurem més endavant que ens interessa no comptabilitat els valors inicials perquè no són fiables encara.

Question 12: When does the simulation end and what makes it finish? Reason your answer.

Answer: El programa acaba quan el temps de la simulació en slots arriba a la durada de la simulació indicada també en slots. Això és així ja que el programa principal és un bucle que es realitza tantes iteracions com indica la variable slots.

PART IV: Understanding of the time-driven simulation model. Analyze the code from the program structure.

Question 13: Describe in your own words how the program advances the execution explaining the structure of the main function of the program and how this main relate the protocol components analyzed in the previous sections so that to get the complete simulation in a sequential program. Explain and justify what the program does before each iteration, at each iteration, and after all iterations, the order it does it, and how many iterations it does.

Answer: El programa main és un bucle d'iteracions d'slots, que és el temps en el programa. El nombre d'iteracions és nslots que és la durada de la simulació. Per tant el bucle avança el temps un slot en cada iteració.

Fora del bucle es fan les inicialitzacions abans, i el produir les estadístiques al final. Dins del bucle cal posar el canal a buit abans de començar amb les estacions, generar els paquets que calgui a les estacions que els hi toqui, executar el protocol per cada estació, i finalment amb el sink rebre el paquet transmès si ha sigut correcte (que aquesta informació està en el canal i s'ha de consultar abans de tornar reinicialitzar el canal) i entregar el ack .

Question 14: Compare the program structures (function main) of the event-driven simulation in practice EDS and time-driven simulation in this one. Comment the differences and similarities (what parts are the same in the two simulations?).

Answer:

1. Els dos programes son dos bucles "infinits" un avançant el temps de simulació i l'altre avançant a la llista d'events. Un para quan s'acaba el temps d'execució i l'altre quan acaba els events.

2. El temps real avança de manera més igual en una simulació orientada a temps (és una constant la majoria de vegades) i de manera més imprevisible en la simulació orientada a events (fa salts en funció dels temps d'execució de cada event).

3. La simulació per events necessita del kernel (o agenda) d'events, mentre que la gestió de temps només és una variable

(rellotge de la simulació) que s'incrementa. Per tant la primera sol fer servir un paquet de simulació (que proporciona el kernel de gestió d'events) mentre que la segona no és necessari.

Question 15: What time unit does this TDS simulation work and advance? How have you identified it?

Answer: l'unitat és l'slot primordialment perquè regeix el bucle de les iteracions. I per tant és la unitat mínima de decisió en la transmissió, és a dir, només passen coses en aquests punts discrets del temps. Per tant no cal tenir dins la simulació una precisió temporal més acurada.

Question 16: The time in the simulation is real time or logical time, global time or local time? Reason your answers

Answer: La simulació només té un rellotge per tant el temps és el mateix en tot el programa i per tant és temps global. No hi ha cap unitat o procés que operi diferent segons una altre rellotge.

La unitat de temps és l'slot i per tant una unitat de temps lògica, però hi ha el càlcul que relaciona el temps lògic amb el temps real. Donada l'ampla de banda del canal podem convertir slots (agrupació de bits) en segons que seria el temps real.

Aquesta relació està incorporada en les definicions següents. MS_TIME que defineix la durada d'un slot i les definicions de canvis d'unitats d'una a l'altre : NORMtoMBPS, MSECtoSLOTS.

Tot i així al començar sempre "a l'hora" 0 sempre serà temps lògic. No correspon a l'hora real (del rellotge de la màquina per exemple).

Question 17: How does the simulation time inside the simulation relate with the time the simulation takes to run that it is indicated in the output of the simulation?

Answer: No hi ha relació. Una cosa és la capacitat de CPU de l'ordinador on s'executa la simulació i per tant un ordinador més potent tardarà menys a executar la simulació que un altre menys potent. L'altre és el temps dins la simulació que identifica quines operacions cal fer i fins quan.

PART V: Learning Summary

Question 18: Final Understanding – What level of comprehension you think you have reached on the operation and implementation of a time driven simulation (TDS)? Briefly summarize the essence of what you have understood (3-5 lines) and indicate your pending doubts in a concrete form so that we can discuss them. Note: this understanding summary is very significant to infer your conceptual understanding. Each member of the team must write its own section without having read the text of the others so that not to condition your writing.

Answer:

Question 19: (Optional) Final Evaluation – Provide a constructive evaluation of the quality of this assignment with respect to meet the learning objectives of the related theory concepts. Comment from all perspectives you feel relevant (motivation, interest, usefulness, difficulty, efficiency, detail, duration, what is missing or too much? Or any other...) to improve it. We encourage you to write your positive and negative comments as all constructive criticism is useful to help improve the learning process.

Answer:

Materials

1. saloha basic program
2. Slides with the protocol slotted-aloha definition (provided with this handout assignment)
3. Theory slides: Simulation Modeling chapter
4. C on-line tutorial: c_standard_library, cprogramming (http://www.tutorialspoint.com) (or your preferred)

Submission instructions

Practice Submissions are not graded. If submitted before deadline indicated in aula global it gives extra grade added to your final practice grade[1]. If (pre)submissions are not fully completed the extra grade will be proportional to the full work to be done in the (pre)submission. This promotes you get the work done when needed to have interactive sessions and for you to consolidate concepts overtime.

The pre-submission can be done and submitted in teams or individually (as you prefer). The final submission must be submitted in groups of 3.

It is highly recommended to do the pre-submission section prior to the practice section. Hence, include the pre-submission section filled (as you have it) in the final submission document.

Optional Pre-submission (P): Answer part II of the document. Rename the handout file with the following naming notation:

- Individual submissions: `SMX_P2TDSP_NIAxxxx_LastName.pdf` where `xxxx` is your NIA value and `LastName` your first surname.
- Team Submissions: SMX_`P2TDSP_GroupX.pdf` where `X` is the letter of your practice team.

Optional Practice (final) submission (Final - F) (in group):

1. Submit all handout full although only sections I, III, IV and V count for the extra grade of the final submission. Name the final submission as follows:

   `SXM_P2TDSF_GroupX.pdf` where `X` is the letter of your practice team.

2. Create a directory named P2TDS-GroupX.
   - Put in the directory all previously mentioned files of the submission (pre-submission if you have and final submission).
3. Compact the directory P2TDS-GroupX with zip/rar/tar and submit a single file named P2TDS-GroupX.tar/zip/rar

The submissions must be done in aula global within the deadlines (pres-submission, submission) indicated in there. **Make sure you put the name inside the file and in the filename of all files (reports, code files, excels, matlab...)** so that the authors can be known when reading the documents (without looking at the directory or files names). The grade will be assigned only to the names that appear in all material.

[1] See evaluation section in pla docent for details