CS 1699: Wireless Network

Project:
CSMA-CA Simple Simulator

1 Goal

Wireless medium is considered as broadcast medium. This means that every wireless station can hear, use the medium any time. This condition can result in collision among multiple transmission and decrease wireless channel utility. Consequently, a scheme for wireless channel access must be in place to manage the access to wireless medium

This project requires every student to develop a simple simulator for medium access for wireless communication. This project aims to have the students gaining better understanding on logic of IEEE 802.11 medium access and its performance as the number of participating stations increases. In this project context, the medium access scheme will follow the steps which are implemented in CSMA-CA (Carrier Sense Multiple Access – Collision Avoidance). Note that although this simulator adopt the logic of CSMA-CA in general sense, but it does not represent precisely the scheme implemented in IEEE 802.11 standard.

2 Programming Language

2.1 General Requirement

This project is not restricted to a particular programming language. However, C/C++ and Java are preferred. Hence, the students are expected to be familiar and hands on with those languages. The use of other language is not prohibited, but should consult the class instructor in advance.

2.2 Specific Requirement

To implement this simulator, the student should be hands on multithreaded programming in their respective language. Furthermore, the knowledge of socket programming is also recommended to implement this project.

3 Project Description

3.1 General

The simulator basically a program that emulates a wireless communication environment in which several wireless stations attempt to access the medium. This simulator can also be implemented as a set of programs that run simultaneously. The choice is up to the student to decide.

For this project, a pre-determine number of wireless station will be given. When a station has message ready to send, it will perform CSMA-CA to access medium before transmitting data. The station should record the time that has been elapsed to accomplish transmission of entire message. If multiple packets of data are needed to accomplish the transmission of entire message, then the station should keep track the total time needed to perform multiple packet transmissions.

In this project, the outcome of the simulation would be the average time needed to accomplish message transmissions in each node, given the activity rate of each stations. Activity rate of a station is defined as the probability of a station to have message ready to send after some interval of time. Thus, a high probability of having message ready to send of a station indicates its high activity on channel access. Thus, in this context, the station will perform more frequent CSMA-CA scheme. This condition results in higher contention among stations when accessing the medium. Consequently, the time needed to complete transmission also increases.

To implement this simulator, each station (which is emulated by the program) should run independently. Each wireless station accesses the medium whenever it needs to, irrespective to other wireless station's state is. However, each station should perform the same logic of CSMA-CA. Implementing such complex steps of CSMA-CA may require students to use multithreaded technique to realize the application, especially if it would be developed as a single large program.

If the simulator were implemented as a set of independent applications, each application represents a wireless station and compete with other station to get access to the medium. A medium will be represented by an application that needs to be invoked together with wireless station applications. Communication between medium application and wireless station may use Socket technique. To run the simulator, student may prepare a small script (batch file or shell script) to execute the programs, instead of running them manually.

3.2 Detailed Description

3.2.1 Multithread programming Technique

This detailed description is relevant for single program implementation, using multithread programming technique. This document may use C style of programming to describe the example of the topic. Other style of programming should be adapted accordingly

Each wireless station would be implemented by a single thread. The number of wireless station, denoted as N, would be 8 units. Each wireless station should implement its own "clock unit" or "tick of time unit". It is not necessarily to be the same as CPU clock. For instance, a student can define 100 ms as 1 tick of time unit for your wireless station. Alternatively, a student may define its own clock unit as a

certain number of *for* loops (for example, 1 tick of unit time may be defined as an execution of 100 times of a single loop). The unit time is denoted as t_S

Figure 1 illustrates the flow of the program for each wireless station whenever it wants to access medium.

The simulator starts with making each wireless station to sleep as long as t_D, where $t_D = 240t_S$. When a wireless station finished the sleep, it checks if it had a message to send (block 2) with probability P where $0 < P < 1$. For example, if $P = 0.5$, then it has 50% chance to have message. You may implement this probability using random number generator. In this context, it is assumed that random number generator has uniform probability. Thus, for example, if a random number generator produces a set of numbers that ranges from 0 to 99, then each number should have probability of 0.01 to be generated every time random number generator function is called. Thus, if a probability of 0.3 is desired, it would be equal with a cumulative probability for a set of number that ranges, for example, from 0 to 29 to be generated. Adopting this technique, a station decides to sleep again if the random generator generated a number outside of desired range.

If a wireless station had message to send, it will check medium every t_S . In a multithreaded programming technique, you may represent a medium as an object that holds a private variable (using object oriented programming technique) or a global variable (without object oriented). However, using a global variable, access to this global variable must be protected under mutual exclusive variable to ensure only one access at a time to the variable, or to a block of process that modifies the variable. Consequently, mutual exclusive variable should be used whenever checking or setting medium status procedures are invoked (especially, when executing block 4,8,11,17 and 20 in Figure 1).

If medium is not busy, then it will wait for t_{DIFS} unit time. In this project, to prevent excessive delay, the student can set t_{DIFS} to some value within following range: $20t_S < t_{DIFS} < 40t_S$,

After wait for t_{DIFS}, the wireless station needs to wait for another $t_{cw} = kW$, where k =1 and W is a random number, where $t_S < W < 2Nt_S$ where N denotes the number of wireless station.

When a station already accessed the medium, it will hold the medium status as long as t_p, where t_p is single packet transmission time. For this project, student can set t_p , where $0.3t_D < t_p < 0.6t_D$. Parameter t_p emulates the time needed to send a DATA packet (block 18)

After finishing sending a single DATA packet, the station is assumed to receive ACK packet within t_{IFS} duration. In this project, $t_{IFS} = 10t_S$. The station who claimed the medium should release the medium after t_{IFS} expired and wait for another t_{DIFS} . This whole process is considered as one cycle in channel access. Total time t_{tot} should be kept recorded during the cycles and should be printed (either to file or standard I/O) for further analysis.

3.2.2 Socket Programming Technique

If the simulator were to be implemented as a set of applications, socket communication may be required. In general, there is no essential difference between wireless station in terms of performing CSMA-CA program flow as shown in Figure 1.

In this context, an additional program is needed to represent medium. When checking medium procedure takes place, wireless station should establish socket connection to medium program (either UDP or TCP). Students can define any port number for this purpose. Because it will be executed in one host (i.e. student laptop), source IP would be the same as destination IP.

To begin checking the medium, wireless station needs to send check status request to medium program. Upon receipt this packet, Medium program should check its channel state variable to see if it is busy. The checking process should be protected using mutual exclusive variable. If it is free, then Medium program should response with OK (if free). Otherwise, Not OK should be sent.

Similarly, setting medium status also begins with wireless station sends request to Medium program. Upon receipt the request, Medium should check its channel state variable under protection of mutual exclusive variable. If by any chance, another wireless station had occupied the channel (although previous check returned OK), then Medium program should response with Not OK. Wireless station should interpret this as "Medium busy" and repeat the channel checking process.

Once Medium program response with OK, it should keep track the most recent wireless station who claim the medium. When the medium wants to set medium to free, the corresponding medium needs to send Reset request to Medium program.

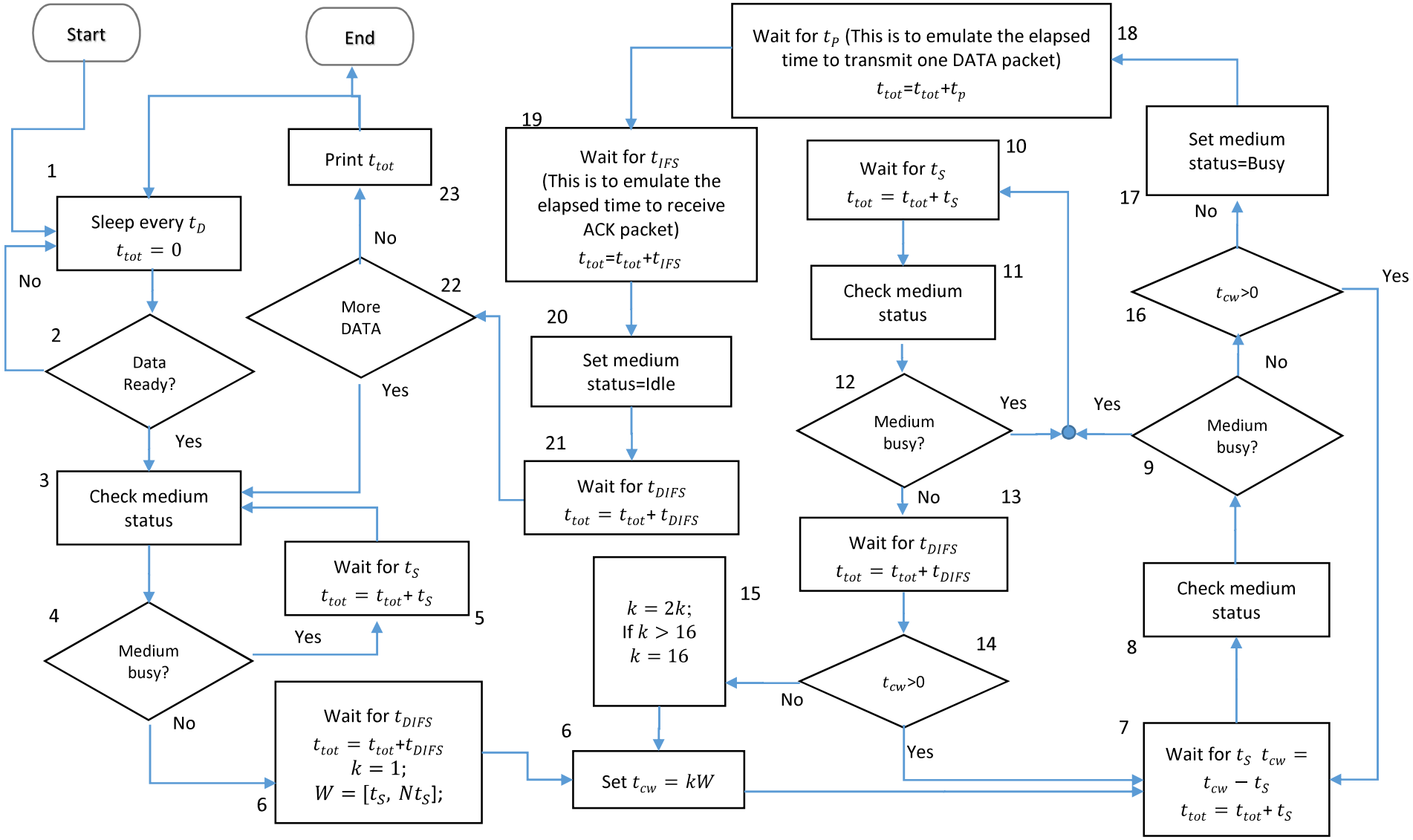
Figure 2 illustrates the process for Check Medium Procedure and Set Medium busy in general.

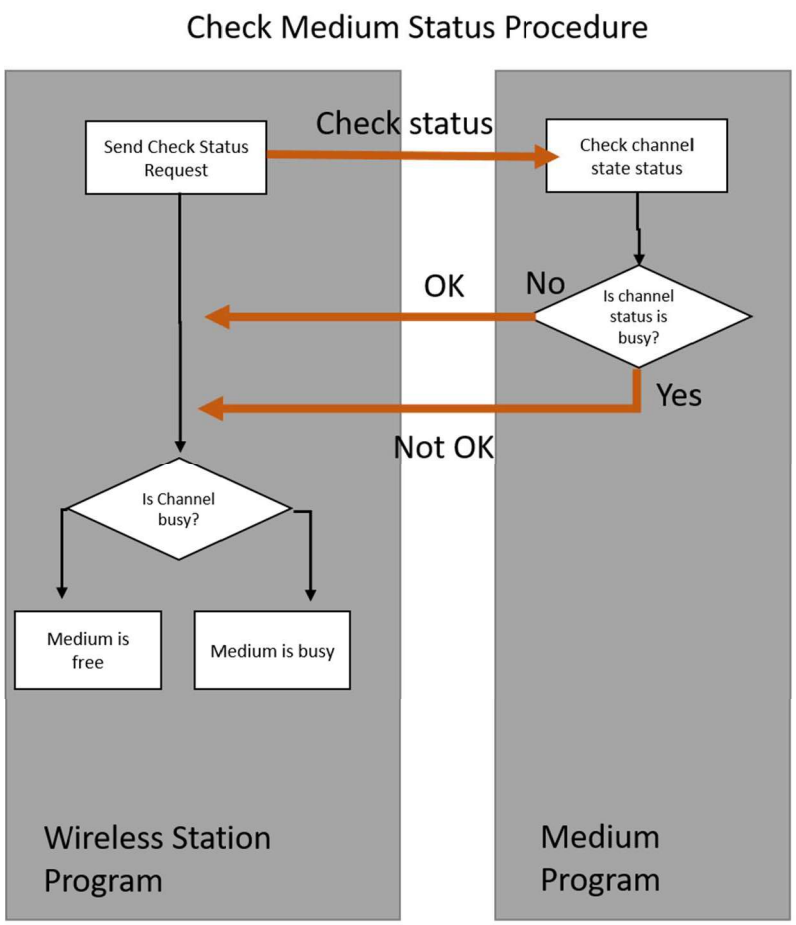
3.3 Simulator Specification

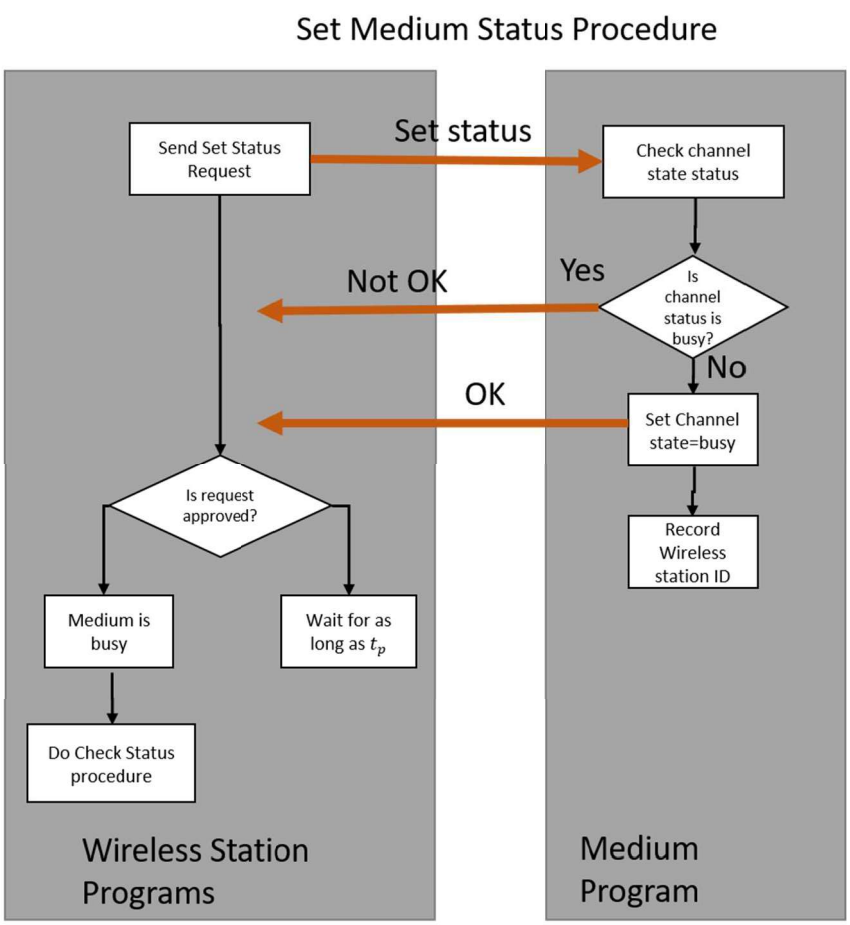
Following table summarize simulator parameters and specification. There are three types of parameters, namely configuration parameter, running parameter, and output. Configuration parameter should be part of simulator's default configuration. Running parameter would be treated as simulator input, which needs to be provided during running time.

Parameter	Notation	Parameter	Value
Number of wireless station	N	Configuration Parameter	8
Unit time	t_S	Configuration Parameter	To be determined by student
Data ready check interval (or sleep duration between Data ready)	t_D	Configuration Parameter	$240t_S$
Probability for data ready for each wireless station	P	Running parameter.	$0 < P < 1$. Since simulator has N stations, then N times of P should be entered to run simulator
Single packet transmission time	t_p	Running Parameter	$0.3t_D < t_p < 0.6t_D$, same for all wireless stations
Duration of DIFS	t_{DIFS}	Configuration Parameter	$20t_S < t_{DIFS} < 40t_S$

Duration of IFS, also implies the duration to receive ACK packet	t_{IFS}	Configuration Parameter	$t_{IFS} = 10t_S$
Total elapsed time	t_{tot}	Output parameter	
Default congestion window	W	Configuration Parameter	$t_S < W < 2Nt_S$. The same value for each wireless station
Number of packet	M	Running Parameter	$1 \leq M \leq 6$. The same value for each wireless station


Start
End
1
Sleep every t_D
$t_{tot} = 0$
2
Data Ready?
No
Yes
3
Check medium status
4
Medium busy?
Yes
No
5
Wait for t_S
$t_{tot} = t_{tot} + t_S$
6
Wait for t_{DIFS}
$t_{tot} = t_{tot} + t_{DIFS}$
$k = 1$;
$W = [t_S, Nt_S]$;
6
Set $t_{cw} = kW$
7
Wait for t_S $t_{cw} = t_{cw} - t_S$
$t_{tot} = t_{tot} + t_S$
8
Check medium status
9
Medium busy?
Yes
No
10
Wait for t_S
$t_{tot} = t_{tot} + t_S$
11
Check medium status
12
Medium busy?
Yes
No
13
Wait for t_{DIFS}
$t_{tot} = t_{tot} + t_{DIFS}$
14
t_{cw}>0
Yes
No
15
$k = 2k$;
If $k > 16$
$k = 16$
16
t_{cw}>0
Yes
No
17
Set medium status=Busy
18
Wait for t_P (This is to emulate the elapsed time to transmit one DATA packet)
$t_{tot} = t_{tot} + t_p$
19
Wait for t_{IFS}
(This is to emulate the elapsed time to receive ACK packet)
$t_{tot} = t_{tot} + t_{IFS}$
20
Set medium status=Idle
21
Wait for t_{DIFS}
$t_{tot} = t_{tot} + t_{DIFS}$
22
More DATA
No
Yes
23
Print t_{tot}


Check Medium Status Procedure
Send Check Status Request
Check status
Check channel state status
OK
No
Is channel status is busy?
Yes
Not OK
Is Channel busy?
Medium is free
Medium is busy
Wireless Station Program
Medium Program


Set Medium Status Procedure
Send Set Status Request
Set status
Check channel state status
Not OK
Yes
Is channel status is busy?
No
OK
Set Channel state=busy
Is request approved?
Record Wireless station ID
Medium is busy
Wait for as long as t_p
Do Check Status procedure
Wireless Station Programs
Medium Program