Goal: Using WSim, a simulator for a simple broadcast network written in Python, develop and experiment with channel access protocols. The pre-lab and lab teach the following important ideas: 1) sharing a broadcast channel using slotted Aloha, carrier sense multiple access (CSMA), and time division multiple access (TDMA); 2) backoff strategies for contention schemes like slotted Aloha and CSMA.

Instructions:

1. Complete the pre-lab activities in Section 4.
2. Complete the lab activities described in Section 5 in lab on Wednesday.
3. Prepare the requested material and think about the questions posed in the Checkoff Sheet, then find a staff member to complete your post-lab interview.

1 Python

For the next three labs (including this one) we'll be using Python, an easy-to-learn interpreted programming language that has a nice selection of data types including support for object-oriented programming, interesting control structures, and a huge (and growing) set of libraries that make it simple to implement almost any processing task.

If you'd like to try out the 6.02 code on your own machine you'll need to install:

Python: http://www.python.org/download/

There are pre-built binaries for Windows, Mac OS X and Linux.

If you'd like to learn or brush up on Python, start with the 6.01 Resources webpage, which has links to several quick tutorials:

http://courses.csail.mit.edu/6.01/spring08/resource.html

For a more comprehensive introduction, we recommend the Python Tutorial written by Python's architect Guido von Rossum:

http://docs.python.org/tut/tut.html
The following sections of the Python Tutorial are particularly useful for 6.02:

Section 3: An Informal Introduction to Python
Section 4.1: if Statements
Section 4.2: for Statements
Section 4.6: Defining Functions
Section 5.1: More on Lists
Section 5.5: Dictionaries
Section 9.3: A First Look at Classes

2 Lab Setup

You can either use the IDLE programming environment to modify and run the code, or you can use python directly. You can start IDLE by running `setup 6.02` and then `idle-602` at the Athena prompt. Use File→Open to open a source file for editing; hit the F5 key to execute the current editor buffer.

Before you start, copy the relevant files over from the course locker to your 6.02 directory (assumed to be in your home directory):

```
athena% setup 6.02
athena% cp -R /mit/6.02/Labs/Lab9 ~/6.02
```

`lab9.py` is the file you need to execute with python.

3 WSim: A Simple Broadcast Network Simulator

This lab uses a simple packet-level network simulator for a broadcast network. You will be writing a small amount of code to develop various channel access protocols and measure how they perform under different conditions.

WSim is split into three files: `wnet.py`, `wnode.py`, and `util.py`. The first two files implement class definitions for `WirelessNetwork` and `WirelessNode`, respectively, and provide a graphical front-end; `util.py` implements class definitions for `Packet` and `Stats`. For the lab, you will mostly need to modify or add code to `wnode.py` alone.

The graphical front-end of WSim is shown in the screenshot in Figure 1. The top part of the window shows the nodes; near each node are three numbers, A (number of packet transmission attempts), S (number of successful packets), and Q (the nodes queue length), which are helpful in understanding what your node is doing. On the side is a menu with various options, both ones you can set and ones you choose. These options are explained below in Section 3.1.

Underneath the palette that shows the nodes is a row of clickable buttons that control the simulation:

- **Reset**: Read the parameter selections and initialize the network and the nodes to these parameters. Reset time to 0 and initialize all statistics.

---

1Our version of IDLE is currently only available under Linux; you will need to come into lab to run it.

---
• **Stats**: Print out various per-node stats at any time during the simulation; they are printed by default at the end of the simulation too. An example for slotted Aloha for the settings shown in the screenshot:

```
Time 10000 attempts 10001 success 3893 coll 6108 util 0.39
Node 0 attempts 637 success 242 coll 395 latency 1 backoffs 0
Node 1 attempts 646 success 246 coll 400 latency 1 backoffs 0
Node 2 attempts 617 success 227 coll 390 latency 1 backoffs 0
Node 3 attempts 595 success 240 coll 355 latency 1 backoffs 0
Node 4 attempts 583 success 224 coll 359 latency 1 backoffs 0
Node 5 attempts 606 success 236 coll 370 latency 1 backoffs 0
Node 6 attempts 606 success 242 coll 364 latency 1 backoffs 0
Node 7 attempts 627 success 246 coll 381 latency 1 backoffs 0
Node 8 attempts 599 success 242 coll 357 latency 1 backoffs 0
Node 9 attempts 655 success 251 coll 404 latency 1 backoffs 0
Node 10 attempts 640 success 249 coll 391 latency 1 backoffs 0
Node 11 attempts 638 success 259 coll 379 latency 1 backoffs 0
Node 12 attempts 636 success 258 coll 378 latency 1 backoffs 0
Node 13 attempts 637 success 234 coll 403 latency 1 backoffs 0
Node 14 attempts 630 success 240 coll 390 latency 1 backoffs 0
Node 15 attempts 649 success 257 coll 392 latency 1 backoffs 0
```

• **Step N**: advance the simulation by N (1, 10, 100) steps or “all”, which runs the simulation to the end. The total simulation time is settable, as explained below.

• **Exit**: Does the obvious thing...
A status bar appears at the bottom of the window with the following information:

- **Time**: The current time-slot.
- **Attempts**: The total number of packets that have been sent by all nodes over the channel so far.
- **Success**: The total number of successful receptions, i.e., packets without collision.
- **Collisions**: Number of collisions so far.
- **Status**: When you move the mouse near a node on the map, the node’s average packet latency and its queue’s contents appear. The average latency is measured over each uncollided packet; it is the average time duration between the time-slot in which packet is sent over the channel and the time-slot in which the packet appeared at the node (the packet’s start time). Each item in the queue is a packet denoted by its start time.

### 3.1 Protocols and Parameters

We will look at three protocols: Slotted Aloha (“S. Aloha”), CSMA, and TDMA. You can pick the right one using the “Protocol” pull-down menu on the left.

The simulation can be controlled by selecting the following parameters (defaults also shown).

- **Num nodes**: The number of nodes sharing the channel (default = 16).
- **Load % (pkts/slot)**: The total offered load by all the nodes in the network, specified as a percentage and in units of “packets per time slot” (default = 100). For example, if the load is 80, then over the simulation, the expected number of time slots that will be “occupied” by a packet from some node is 0.8. It is possible to set the offered load to more than 100 (the maximum allowed is 1000, which is an offered load $10 \times$ higher than the channel’s transmission rate. One of your tasks below will be to measure utilization as a function of offered load for various protocols.
- **Pkt time (slots)**: The number of time-slots occupied by a packet; i.e., the packet size in time-slot units (default = 1). All packets in any given simulation run are the same size. When the packet size is large, the corresponding channel load is scaled by the packet time.
- **Sim time (slots)**: Total simulation time (default = 10000). It’s unlikely that you’ll need to change that much unless the load is much smaller than 100%, the number of nodes is much larger than 100, or the packet time (size) is large.
- **Protocol**: Pick one of “S. Aloha”, “CSMA”, or “TDMA”. You’ll need to write the code for each of these channel access schemes. (Default setting is “S. Aloha”.)
- **Retry**: “No” (the default) or “Yes”. For “TDMA”, since collisions can’t happen, this option is always “No”.

4
Backoff: What kind of backoff protocol do you want? This option is meaningful only when “Retry” is “Yes”. The default setting is “Bin. expo”, or binary exponential backoff, whose code you’ll have to write. We’ve also provided an option for “Mine”, where you can invent anything you want, and “None” for no backoffs (this choice will perform abysmally when Retry is “Yes!”).

The backoff option has no effect for the TDMA protocol; it is meaningful only for the contention protocols (S. Aloha and CSMA).

Src skew: The default option, “No”, takes the entire offered load divides it equally amongst the nodes. If skew is “Yes”, then the load, $\ell$ is divided in geometrically-spaced amounts. Node 0 presents a load of $\ell/2$, node 1 $\ell/4$, and so on. The last two nodes each present the same load, $\ell/2^{n-1}$, where $n$ is the total number of nodes.

**Note well:** After you set the options you want, make sure to hit Reset for the settings to take effect, before running the simulation.

A word on how the nodes generate traffic is in order. Each node takes its allocated rate and generates traffic independently from an exponential distribution whose mean is the specified rate (pkts/time-slot). A very nice property of this source distribution is superposition: the sum of independent exponentially-distributed data sources (also called Poisson sources) is also Poisson, and the rate of the summed distribution is the sum of the individual rates.

This lab won’t be concerned with modifying the source distribution, other than changing the offered load and setting skew. When you run WSim or select “Reset”, a set of lines of the form “attaching 0.0625 to Node N” appear. These lines tell you the average per-node offered load of the configuration, in packets/time-slot.

### 3.2 How WSim Works

The high-level goal is to understand how different protocols perform under various parameter settings and comparing them. WSim operates one time-step at a time. In each time-step the step method in the WirelessNetwork class runs the following steps:

1. Check for collisions on the broadcast channel, and if so, set a flag on each packet currently on the channel.
2. Wrap up any transmissions that ended in this time slot and update statistics.
3. Call each node’s step function. If the node is already not transmitting a packet, and the time is larger than nextslot (set by the backoff scheme), then the node transmits the enqueued packet at the top of the queue whose start time has come. The key check here is the call to the node’s channel_access function. This function returns True if, and only if, the node can start transmitting according to the channel access protocol.

   The node’s step method also checks if new packets need to be generated.
4. Increment the time-slot and start up any new transmissions.

In the lab, your main (but not only) task will be to implement the node’s channel_access method in wnode.py. That involves filling in the code for Aloha, CSMA, and TDMA.

In addition, you will implement the code for backoff in the same WirelessNode class.

The pre-lab exercises are on the next page.
4 Pre-lab Exercises

Please complete the following tasks individually before you come to lab on Wednesday.

Copy over the relevant files for the lab and run it. You’ll see a picture like the screenshot picture shown before.

1. Write the code for slotted Aloha in the method `Aloha` in the `WirelessNode` class. (This is straightforward and there are no tricks involved!) Demonstrate your code running at the beginning of the lab to one of the staff members. Pay attention to the utilization. Does it make sense given what you learned in class?

2. Plot a graph of utilization v. offered load for slotted Aloha and show it to one of the staff members at the beginning of the lab. Vary the offered load from 10% to 1000% (in suitable steps—you don’t need to do it in steps of 0.1 exhaustively, but can pick your load settings more cleverly keeping in mind what the curve is likely to look like).

3. Can you set a suitable parameter or pick a particular choice from the options on the left sub-window to make your slotted Aloha perform similarly to unslotted Aloha, for any number of nodes? Show the parameter and explain to a staff member why your parameter setting mimics unslotted Aloha.

4. Read and understand the following description of CSMA (carrier sense multiple access). CSMA is an attempt to improve on the performance of slotted Aloha. As in slotted Aloha, time is slotted, but unlike in slotted Aloha, before a node sends, it listens on the channel. If another transmission is already in progress, then the node defers its transmission. (In WSim, the WirelessNetwork class has a channel variable that is set to busy whenever some node is sending data.) A node sends data only if at the beginning of a time slot it finds that the channel is not busy.

5. From the lecture notes, make sure you understand how TDMA (time division multiple access) works.

The lab exercises are on the next page.
5 In the Lab (Wednesday, April 30, 2008)

1. **CSMA.** Implement CSMA in WSim.
   
   (a) When the pkt time is 1, CSMA performs the same as slotted Aloha. How come?
   
   (b) Plot a graph showing the utilization of CSMA when the packet time is 10, and slotted Aloha when the packet time is 1, for different offered loads. Does your graph make intuitive sense?

   Get the “CSMA” part of your checkoff sheet signed.

2. **TDMA.** Implement TDMA in WSim. It might be best to do it in two parts, first assuming that the packet time is 1 slot and making sure your code does the right thing, then modifying that code to handle packet times that are longer than 1 time slot.

   (a) Make sure your TDMA scheme has no collisions and exhibits round-robin behavior.

   (b) TDMA has no collisions, so why might we not want it in general? Using the parameter settings in the left sub-window, pick settings that demonstrate higher utilization for both CSMA and slotted Aloha.

   Demonstrate TDMA to a staff member and get that part of the checkoff sheet signed.

3. **Backoff.** Reset the protocol to S. Aloha and a pkt time of 1. Choose the “Yes” option for “Retry”. When you run it, the utilization is 0. You need a backoff scheme.

   (a) Binary exponential backoff is the scheme discussed in class, where each node maintains a *exponential backoff window*, \( W \). On a collision, rather than retrying immediately, the node picks a random integer in \((0, W)\) and tries after that much more time. Here, 0 picks the slot *following the one in which the collision occurred*. If the packet succeeds, then \( W \) is set to 1, or to some minimum value of your choosing. If, on the other hand, a collision occurs, then \( W \) is doubled. To implement binary exponential backoff, construct an array, \( \text{expo_backoff_window} \), which has the set of possible exponential backoff values in the WirelessNode \( \text{init} \) method. Then, the protocol only has to keep track of the correct index into this array, \( \text{backoff_level} \), updating it both each time \( \text{backoff()} \) is called and when a transmission succeeds. In Python, \( \text{random.random()} \) returns a random number between 0 and 1. The \( \text{backoff()} \) method should update \( \text{nextslot} \), a variable in the WirelessNode class. It maintains the next time slot that the node is allowed to send a packet.

   (b) Run your backoff protocol at two or three interesting load settings for S. Aloha and CSMA. How does the utilization of your backoff with Retry “Yes” compare to the corresponding S. Aloha and CSMA settings when Retry is “No”?

   (c) What can you say about the distribution of attempts and successful packet transmission across the different nodes? Why does that happen?

   (d) Can you think of a way to alleviate the problem observed in the previous question? Time permitting, try to implement a modification to the backoff scheme, or develop a different kind of backoff.

   Get the Backoff part of the checkoff sheet signed by a staff member.
Check-off sheet for Lab #9 (April 30, 2008)

Names: 

1. Pre-lab checkoff (before lab, for each group member by initials):
2. CSMA: 
3. TDMA: 
4. Backoff: 

Interview questions:

1. The theoretical analysis for slotted Aloha predicted a maximum utilization of \( \frac{1}{e} \). For the default settings of WSim, the number is slightly higher. What theoretical assumptions cause the difference?

2. For the “perfect” broadcast network studied in this lab, can CSMA ever have a worse utilization than slotted Aloha?

3. Suppose the skew option is “No”, i.e., all nodes generate data at the same rate. Suppose you run slotted Aloha with Retry set to “Yes” using exponential backoff. How does the utilization of this version of slotted Aloha compare with TDMA? Answer the question for low offered loads (say less than 30%) and high load (say over 60%).
   And how does the latency for this version of slotted Aloha compare with TDMA?

4. What weaknesses does your exponential backoff protocol have? Can you think of a way to overcome them?