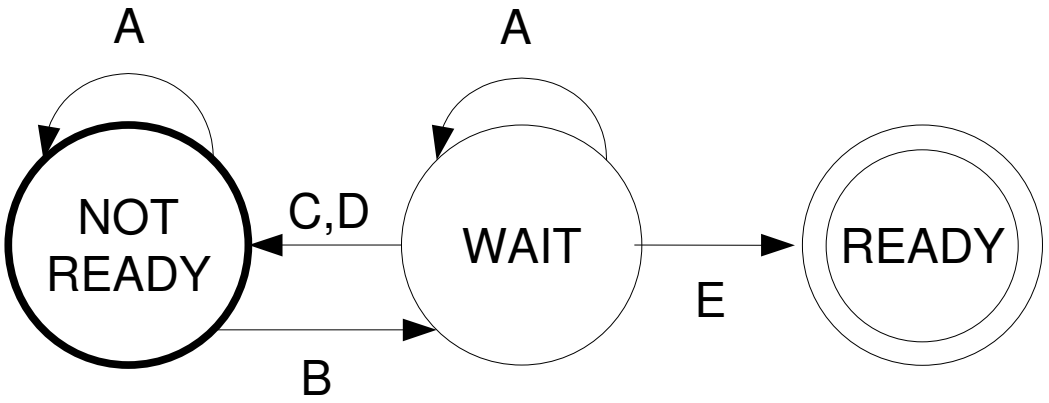
LooseMAC Simulator

Brett D. Estrade
CSC 7501, Spring 2007
estrabd@lsu.edu

LooseMAC Finite State Machine

- Based on my understanding:



- A: heard beacon / manage vector listing (includes scheduling error msg if needed)
- B: sent beacon / mark current time + lambda as time to be ready
- C: heard error message / pick new sigma, schedule beacon msg
- D: detected collision / action for C + schedule to send an conflict report (msg will always be <1,1> when sending after a D transition).
- E: survived waiting period / -

Simulator Highlights - Features

- Can simulate an arbitrary topology and set default values for frame size and slots assigned
- Slots not explicitly set will be chosen by random
- Actions driven by FSM
- Messages passed using a mailbox delivery scheme, so colliding messages are corrupted during the actual sending
- Collisions, marking conflicts, and beacon messages are handled accordingly based on the state of the receiver

Graph Input

- Networks can be defined using a the traditional graph format

```
numNode [Λ]    # comments allowed
node0 (numAdjacent) n1 n2 ... nnumAdjacent [slots..]
```

- LAMBDA (frame size) can be specified explicitly as the second value of the first line
- For each node, an optional set of slots may be explicitly set; this is useful for testing out different starting conditions

Simulator Steps

1. Time step incremented
2. All nodes using current slot send messages if either of the send_beacon or send_error flags are set; mailboxes receiving more than one message per time step get set with a “corrupted” message
3. All nodes check their mailbox slots and react accordingly based on the message type and their own state
4. At the end of a time step, all nodes meeting the criteria are made ready
5. Time steps continue until all nodes are ready or until all nodes are either ready or are not ready, but are surrounded by ready nodes (more on this in a bit)

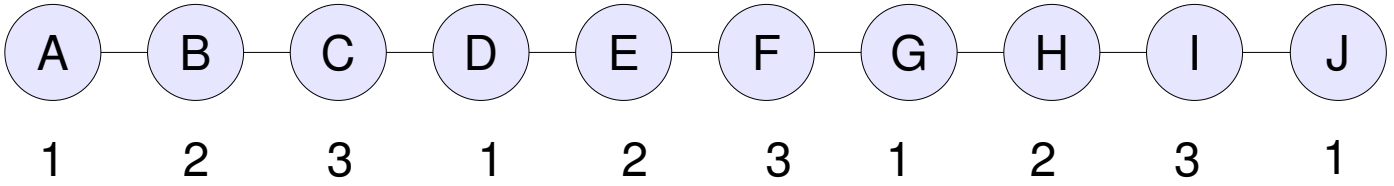
Simulator Caveats

- Simulation is for synchronized frames and slots over all nodes only; there is not handling of off sets, but this might be done easily with a simple translation function/table
- Tested well, but there could be some issues with how I am simulating things
- I may have missed a detail somewhere
- The code is kind of messy, and I am sure there is a better way to program such a thing – but I did learn a lot
- The simulation is a sequential emulation of distributed, concurrent actions – the simulation should be programmed this way, too
- The case where 2 adjacent nodes sharing the same channel is not handled, but I think it could be given a small tweak in how the nodes track slots

Issues

- When in the loop should states be checked for being ready?
- A lot of assumptions were made, including
  - when nodes were made ready – currently, it is at the end of the time step
  - READY nodes do not send any messages, ever
  - nodes can transmit more than once per frame

An Ideal Case



- Ideal means that there are no conflicts, so all nodes are READY in frame 2
- Simulator handles this as expected, which is that all nodes are ready by slot 3 in frame 2 (6 time steps total)

An Ideal Case

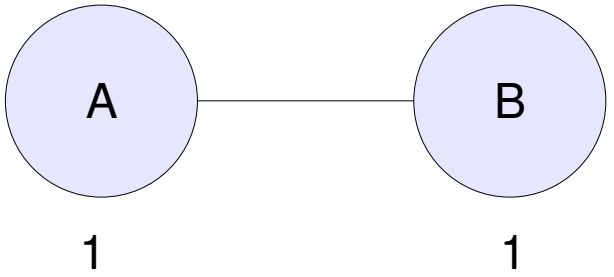
```
[ t=1 (slot=1) ]
==================
  Event! 6: sent BEACONMESG to: [5,7] and will be ready @ next slot 1 (t=4)
  Event! 3: sent BEACONMESG to: [2,4] and will be ready @ next slot 1 (t=4)
  Event! 9: sent BEACONMESG to: [8] and will be ready @ next slot 1 (t=4)
  Event! 0: sent BEACONMESG to: [1] and will be ready @ next slot 1 (t=4)
  Event! 7: heard beacon from 6
  Event! 2: heard beacon from 3
  Event! 8: heard beacon from 9
  Event! 1: heard beacon from 0
  Event! 4: heard beacon from 3
  Event! 5: heard beacon from 6
Nodal State Report [0 / 10 done]:
  Node 0: <WAITING> @ slot using slot 1; Mail Queue: BEACON->[0] CONFLICTREPORT->[0]
  Node 1: <NOTREADY> @ slot using slot 2; Mail Queue: BEACON->[1] CONFLICTREPORT->[0]
  Node 2: <NOTREADY> @ slot using slot 3; Mail Queue: BEACON->[1] CONFLICTREPORT->[0]
  Node 3: <WAITING> @ slot using slot 1; Mail Queue: BEACON->[0] CONFLICTREPORT->[0]
  Node 4: <NOTREADY> @ slot using slot 2; Mail Queue: BEACON->[1] CONFLICTREPORT->[0]
  Node 5: <NOTREADY> @ slot using slot 3; Mail Queue: BEACON->[1] CONFLICTREPORT->[0]
  Node 6: <WAITING> @ slot using slot 1; Mail Queue: BEACON->[0] CONFLICTREPORT->[0]
  Node 7: <NOTREADY> @ slot using slot 2; Mail Queue: BEACON->[1] CONFLICTREPORT->[0]
  Node 8: <NOTREADY> @ slot using slot 3; Mail Queue: BEACON->[1] CONFLICTREPORT->[0]
  Node 9: <WAITING> @ slot using slot 1; Mail Queue: BEACON->[0] CONFLICTREPORT->[0]
State tracking queues
NOTREADY 7,2,8,1,4,5
WAITING  6,0,3,9
READY
```

First time step

```
[ t=6 (slot=1) ]
==================
  Event! 2: is ready @ time 6 (slot 1)
  Event! 8: is ready @ time 6 (slot 1)
  Event! 5: is ready @ time 6 (slot 1)
Nodal State Report [10 / 10 done]:
  Node 0: <READY> @ slot using slot 1; Mail Queue: BEACON->[0] CONFLICTREPORT->[0]
  Node 1: <READY> @ slot using slot 2; Mail Queue: BEACON->[0] CONFLICTREPORT->[0]
  Node 2: <READY> @ slot using slot 3; Mail Queue: BEACON->[0] CONFLICTREPORT->[0]
  Node 3: <READY> @ slot using slot 1; Mail Queue: BEACON->[0] CONFLICTREPORT->[0]
  Node 4: <READY> @ slot using slot 2; Mail Queue: BEACON->[0] CONFLICTREPORT->[0]
  Node 5: <READY> @ slot using slot 3; Mail Queue: BEACON->[0] CONFLICTREPORT->[0]
  Node 6: <READY> @ slot using slot 1; Mail Queue: BEACON->[0] CONFLICTREPORT->[0]
  Node 7: <READY> @ slot using slot 2; Mail Queue: BEACON->[0] CONFLICTREPORT->[0]
  Node 8: <READY> @ slot using slot 3; Mail Queue: BEACON->[0] CONFLICTREPORT->[0]
  Node 9: <READY> @ slot using slot 1; Mail Queue: BEACON->[0] CONFLICTREPORT->[0]
State tracking queues
NOTREADY
WAITING
READY    6,3,7,9,2,8,1,4,0,5
==================
all ready in 6 timesteps
```

Last time step (t=6, σ=3)

Simple Conflict



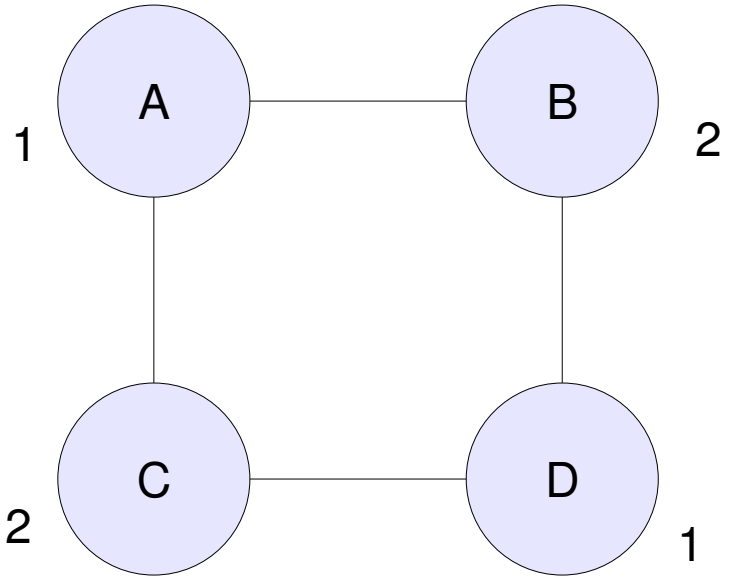

```
2 [2]
0 (1) 1 [1]
1 (1) 0 [1]
```

- Resolves itself quickly, even for a small frame size
- Currently the simulation does not treat this as a legal case, though I believe a very minor modification would allow this

Simple Conflict

```
default lambda set to: 10
2 nodes in network
lambda set to: 2
  Event! 0: slot set EXPLICITLY to 1
  Event! 1: slot set EXPLICITLY to 1
[ t=1 (slot=1) ]
==================
  Event! 1: sent BEACONMESG to: [0] and will be ready @ next slot 1 (t=3)
  Event! 0: sent BEACONMESG to: [1] and will be ready @ next slot 1 (t=3)
  Event! 1: heard beacon from 0
  Event! 1: detected marking conflict!
  Event! 0: heard beacon from 1
  Event! 0: detected marking conflict!
Nodal State Report [0 / 2 done]:
  Node 0: <WAITING> @ slot using slot 1; Mail Queue: BEACON->[0] CONFLICTREPORT->[1]
  Node 1: <WAITING> @ slot using slot 1; Mail Queue: BEACON->[0] CONFLICTREPORT->[1]
State tracking queues
NOTREADY
WAITING  1,0
READY
==================
[ t=2 (slot=2) ]
==================
Nodal State Report [0 / 2 done]:
  Node 0: <WAITING> @ slot using slot 1; Mail Queue: BEACON->[0] CONFLICTREPORT->[1]
  Node 1: <WAITING> @ slot using slot 1; Mail Queue: BEACON->[0] CONFLICTREPORT->[1]
State tracking queues
NOTREADY
WAITING  1,0
READY
==================
[ t=3 (slot=2) ]
==================
  Event! 1: is ready @ time 3 (slot 2)
  Event! 0: is ready @ time 3 (slot 2)
Nodal State Report [2 / 2 done]:
  Node 0: <READY> @ slot using slot 1; Mail Queue: BEACON->[0] CONFLICTREPORT->[1]
  Node 1: <READY> @ slot using slot 1; Mail Queue: BEACON->[0] CONFLICTREPORT->[1]
State tracking queues
NOTREADY
WAITING
READY    1,0
==================
all ready in 3 timesteps
```

Slightly More Complicated Conflict




```
4 [4]
0 (2) 1 3 [1]
1 (2) 0 2 [2]
2 (2) 1 3 [1]
3 (2) 0 2 [2]
```

- Convergence is extremely sensitive to the frame size
- A frame size of lambda converges consistently in under 300 time steps
- A frame size of 4 converges very rarely – though often within the first 30 time steps when it does;
- more often than not, the simulation runs and runs, the longest being for an hour (~7 million time steps) before I killed the process

Summary

- Basic simulations done thus far are not sufficient to make a judgment about LooseMAC
- More testing must be done, but I think that the simulator work based on the information available in the paper
- I'd like to work on it a bit more to convince myself that it does indeed work
- Although no new details were revealed to me, the code provides a basis for testing details as they are either revealed or understood more clearly
- I will be putting up the code and some input files, and will email the class with the URL