

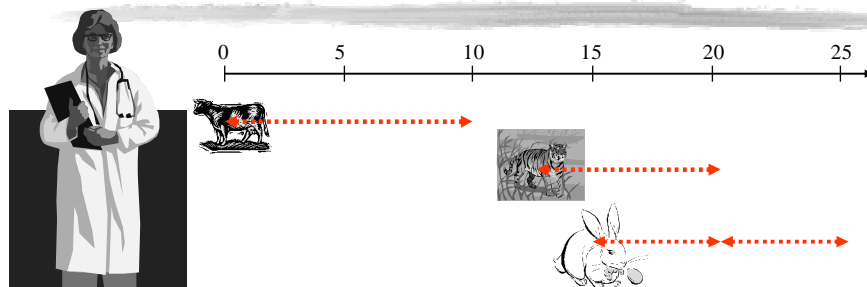
EEC 687/787 Mobile Computing (Spring, 2009)

Ns-2 Laboratory #4 & #5

Prof. Chansu Yu

<http://academic.csuohio.edu/yuc/>
c.yu91@csuohio.edu

Quiz



- (1) Throughput =
- (2) Delay =
- (3) Service time =
- (4) Queueing delay =
- (5) Utilization =

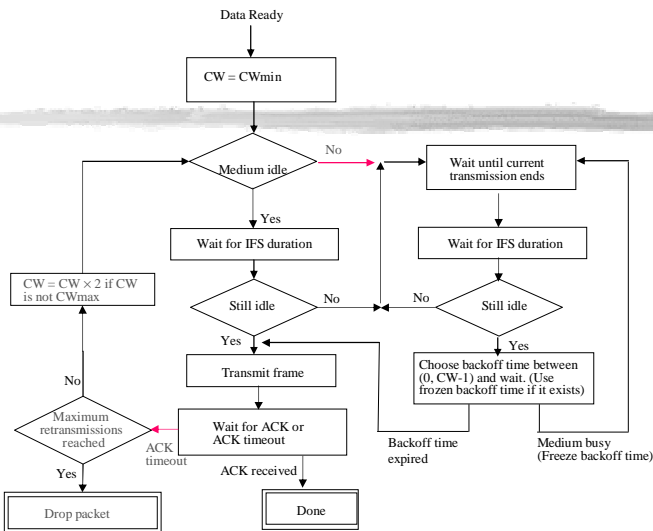
Effect of CW on Network Performance

- ❑ Binary Exponential Backoff (BEB) in 802.11 MAC
 - Since the number of nodes attempting to transmit simultaneously may change with time, some mechanism to manage congestion is needed

- ❑ IEEE 802.11 DCF: Congestion control achieved by dynamically choosing the contention window cw
 - When a node fails to receive CTS (ACK) in response to its RTS (Data), it increases the contention window: cw is doubled (up to an upper bound)
 - When a node successfully completes a data transfer, it restores cw to CW_{min}

3

c.yu91@csuohio.edu



Congestion control

4

c.yu91@csuohio.edu

Effect of CW on Network Performance (in-class lab)

- ❑ In this lab, we will consider a variant of 802.11 where the contention window size is fixed, i.e., $CW_{Min}=CW_{Max}=CW$.
- ❑ We will not dynamically change CW, but still use randomization.
- ❑ We will need a topology under which to study the effect of CW sizes. For this, let us choose a single hop network where all nodes are in range of each other. Specifically we consider a 150m X 150m area. All nodes are involved in two Constant Bit Rate (CBR) conversations: one as source, and one as destination.

5

c.yu91@csuohio.edu

Effect of CW on Network Performance (in-class lab)

1. Get the tcl script `cwfixed.tcl`.
2. In the script you will find the following lines that set the values of `CWMin` and `CWMax` to the desired value:

```
$val(mac) set CWMin_ 31  
$val(mac) set CWMax_ 31
```
3. Usage:

```
> ns cwfixed.tcl -rlen 2
```

It creates a topology of $rlen^2$ nodes arranged in a regular grid (150m x 150m grid area)

6

c.yu91@csuohio.edu

Effect of CW on Network Performance (in-class lab)

4. Run the script for $r_{len}=3, 4$ and 5 (i.e. number of nodes= $9, 16, 25$), and vary $CW_{min}=CW_{max}$ to take the following set of values: $2, 7, 15, 31, 63, 127$ (A total of 18 combinations).
5. Obtain and plot the aggregate CBR throughput and Packet Delivery Ratio (PDR) vs CW size for different r_{len} values.
6. Use “grep” to calculate PDR. To get the throughput, since all flows are a CBR, with a constant packet size of 512 bytes, just multiply number of packets received with 512×8 and divide by total simulation time (25 seconds for this example).

7

c.yu91@csuohio.edu

MILD Algorithm

- ❑ When a node fails to receive CTS in response to its RTS, it multiplies cw by 1.5
 - Similar to 802.11, except that 802.11 multiplies by 2
- ❑ When a node successfully completes a transfer, it reduces cw by 1
 - Different from 802.11 where cw is restored to Cw_{min}
 - In 802.11, cw reduces much faster than it increases
 - MILD: cw reduces slower than it increases
 - Exponential Increase Linear Decrease
- ❑ MILD can avoid wild oscillations of cw when congestion is high

8

c.yu91@csuohio.edu

MILD Algorithm (Lab report #4)

- ❑ In this lab, we will consider that
 - CW increases by 50% upon a failure (MILD-1)
 - CW decreases by one upon a success (MILD-2)

- ❑ How to implement MILD?
 - Modify ns-802_11.cc file

9

c.yu91@csuohio.edu

Implementing MILD-1

- ❑ When a node fails to receive CTS in response to its RTS, it multiplies cw by 1.5
 - Similar to 802.11, except that 802.11 multiplies by 2

- ❑ Variable for contention window: $cw_$

- ❑ How to increase: `inc_cw()`, defined in `mac-802_11.h`
- ❑ When is it called?

10

c.yu91@csuohio.edu

Implementing MILD-2

- ❑ When a node successfully completes a transfer, it reduces cw by 1

- ❑ How to decrease: `rst_cw()`, defined in `mac-802_11.h`

- ❑ When is it called?

- `RetransmitRTS()`, all retransmissions failed
- `RetransmitDATA()`, all retransmissions failed
- `rcvACK()`

} *Move on to
the new packet*

11

c.yu91@csuohio.edu

MILD Algorithm

- ❑ Use the same network environment.

- A single hop network in a 150m X 150m area
- Number of nodes=9, 16, 25
- The same traffic as in `cwfixed.tcl`

- ❑ Obtain and plot the aggregate CBR throughput and PDR vs CW size for different number of nodes. Repeat the same set of simulations for the original 802.11 MAC.

- ❑ Discussion

- What trends do you observe? Which one is better?
- Can you predict what would happen if you try a heavier and a lighter condition?

12

c.yu91@csuohio.edu

13

c.yu91@csuohio.edu

Mobile Network Simulation

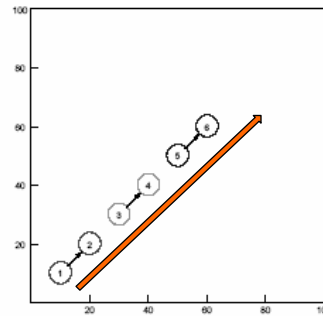
- Large number of mobile nodes (e.g. 50)
- Node mobility
- Traffic

14

c.yu91@csuohio.edu

Mobile Network Simulation

- Number of nodes
- Moving range
- Initial positions
- Moving pattern
 - Direction
 - Velocity
 - Acceleration



- Node WT1 moves towards (90, 90) with speed 20m/s at time 15
`$ns_ at 15.0 "$WT(1) setdest 90.0 90.0 20.0"`

15

c.yu91@csuohio.edu

In-class Lab

- Introduce mobility and draw a throughput chart using xgraph (WT1-> WT2)
 - `$ns_ at 15.0 "$WT(1) setdest 90.0 90.0 20.0"`
 - Does the mobility change the performance?
- Increase the network size to (1000x1000) and repeat the same experiment
 - `$ns_ at 15.0 "$WT(1) setdest 900.0 900.0 20.0"`
 - Does the mobility change the performance?
 - If different than the above, why is that?

16

c.yu91@csuohio.edu

Mobility Model in ns-2

□ RWP (Random Waypoint) mobility model

- The most popular in simulation studies

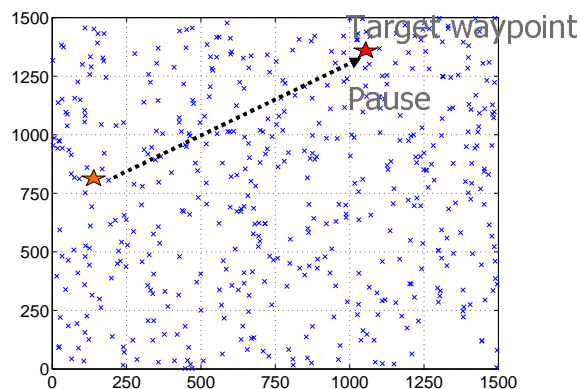
- Two parameters: Pause time and speed $[0, v_{\max}]$

- Pause ▶
- Select a waypoint randomly ▶
- Move toward it with a randomly chosen speed between $[0, v_{\max}]$ ▶
- Repeat

17

c.yu91@csuohio.edu

RWP Mobility Model



18

c.yu91@csuohio.edu

setdest

A Node-Movement Generator

- Generating idea**
Node moves randomly. (distribution of nodes: uniform)
- Location**
~ns/indep-utils/cmu-scen-gen/setdest/setdest{.cc; .h}
- Command format (version 1)**
setdest -v 1 [-n ##] [-p ##] [-M ##] [-t ##] [-x ##] [-y ##]
- Option explanation**
n: number of nodes; p: pause time; M: maximum speed;
t: simulation time; x: maximum x; y: maximum y
- setdest -v 1 -n 25 -p 20 -M 5 -t 100 -x 850 -y 300 > mob

In run script

```
set opt(sc) "./mob"  
source $opt(sc)
```

19

c.yu91@csuohio.edu

cbrgen.tcl

A CBR Traffic Generator

- What is CBR?**
Constant Bit Rate.
- Generating idea**
Randomly pick up node pairs as sources and destinations.
- Traffic generator**
~ns/indep-utils/cmu-scen-gen/cbrgen.tcl
- Command format**
ns cbrgen.tcl [-type ##] [-nn ##] [-seed ##] [-mc ##][rate ##]

20

c.yu91@csuohio.edu

cbrgen.tcl (cont.)

Option explanation

type: traffic/connection type. Must be tcp or cbr.

nn: number of nodes.

seed: seed for generating random number. It is used to generate the random starting time of the traffic.

mc: maximum number of connections.

rate: packet rate = 1 / packet interval

Generate real random traffic

cbrgen.tcl does not generate the real random traffic.

21

c.yu91@csuohio.edu

cbrgen.tcl

A CBR Traffic Generator

For 25 nodes, 10 connections and packet rate of 20

```
ns cbrgen.tcl -type cbr -nn 25 -seed 1 -mc 10 -rate 20 > traffic
```

Read “traffic” file

Read “cbrgen.tcl”

- How to determine packet size?
- How to correctly make 10 connections?
- When does each source start transmitting?
- Who are the destinations?

22

c.yu91@csuohio.edu

Lab Exercise (Lab report #5)

- Download mobility.tcl
- Create mobility file “mob”
- Create traffic file “traffic”

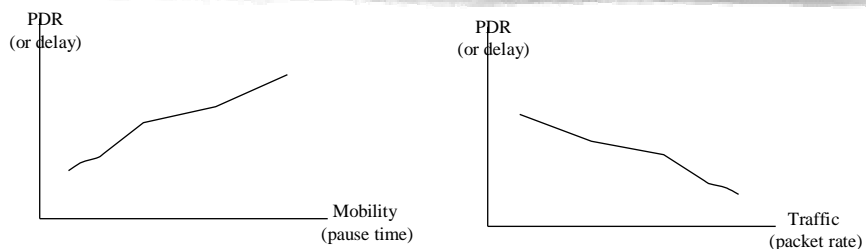
- Run ns-2 to get the trace file and to get average delay and pdr
 - ns mobility.tcl -scen mob -tfc traffic

- To see the effect of mobility (pause or speed), we’ll prepare a number of mobility files such as mob0, mob1, mob2, etc.
 - ns mobility.tcl -scen mob1 -tfc traffic
 - ns mobility.tcl -scen mob2 -tfc traffic
 - ...

23

c.yu91@csuohio.edu

Lab Exercise (Lab report #5)



- Change pause time (0, 10, 50, and 100) when generating “mob” and draw a chart that shows “pause time” versus “pdr/delay”

- Change packet rate (5, 10, 20, 30) when generating “traffic” and draw a chart that shows “pause time” versus “pdr/delay”

24

c.yu91@csuohio.edu

Confidence in Simulation Study

- ❑ Do you see a clear pattern in the simulation results?
 - Noise and randomness in experiments
- ❑ How to eliminate the noise/randomness?
 - Repeat the simulation and get average performance
 - `ns cbrgen.tcl -type cbr -nn 25 -seed 1 -mc 10 -rate 20 > traffic1`
 - `ns cbrgen.tcl -type cbr -nn 25 -seed 3 -mc 10 -rate 20 > traffic3`
 - ...
- ❑ How many do we need to repeat? How much are we confident on what we have measured?
 - PDR measures: 80%, 95% and 65% => average 80%
 - PDR measures: 78%, 82% and 80% => average 80%