EEC 687/787 Mobile Computing (Spring, 2007)

Ns-2 Laboratory (Mar. 5)

Chansu Yu

Cleveland State University

Lab Exercise: Multirate Control

- Multirate support
  - 802.11 supports 1 and 2 Mbps
  - 802.11b supports 1, 2, 5.5 and 11 Mbps
  - 802.11a supports 6, 9, 12, 18, 24, 36, 48 and 54 Mbps
- Advantages of high-rate communication
  - Faster delivery (less delay)
  - More throughput
  - Less energy
  - Issues ???

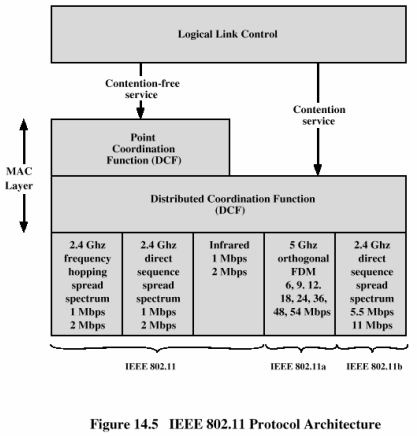


Figure 14.5 IEEE 802.11 Protocol Architecture

c.yu91@csuohio.edu

IEEE 802.11/a/b Physical layer

Table 14.4 IEEE 802.11 Physical Layer Specifications

(a) Direct sequence spread spectrum

Data rate	Chipping code length	Modulation	Symbol rate	Bits/symbol
1 Mbps	11 (Barker sequence)	DBPSK	1 Msps	1
2 Mbps	11 (Barker sequence)	DQPSK	1 Msps	2
5.5 Mbps	8 (CCK)	DBPSK	1.375 Msps	4
11 Mbps	8 (CCK)	DQPSK	1.375 Msps	8

(b) Frequency-hopping spread spectrum

Data rate	Modulation	Symbol rate	Bits/symbol
1 Mbps	Two-level GFSK	1 Msps	1
2 Mbps	Four-level GFSK	1 Msps	2

(c) Infrared

Data rate	Modulation	Symbol rate	Bits/symbol
1 Mbps	16-PPM	4 Msps	0.25
2 Mbps	4-PPM	4 Msps	0.5

(d) Orthogonal FDM

Data rate	Modulation	Coding rate	Coded bits per subcarrier	Code bits per OFDM symbol	Data bits per OFDM symbol
6 Mbps	BPSK	1/2	1	48	24
9 Mbps	BPSK	3/4	1	48	36
12 Mbps	QPSK	1/2	2	6	48
18 Mbps	QPSK	3/4	2	96	72
24 Mbps	16-QAM	1/2	4	192	96
36 Mbps	16-QAM	3/4	4	192	144
49 Mbps	64-QAM	2/3	6	288	192
54 Mbps	16-QAM	3/4	6	288	216

3

c.yu91@csuohio.edu

Lab Exercise: Multirate Control

- A higher rate packet
  - Travels less distance
  - More subjective to interference
- In terms of radio technology
  - Less communication distance means to require a signal strength (receive sensitivity)
  - More vulnerability to interference means to require a higher SNR for successful communication (capture threshold)

4

c.yu91@csuohio.edu

Lab Exercise: Multirate Control

- mac/wireless-phy.cc

```
int WirelessPhy::sendUp(Packet *p) {
…
if (Pr < CSThresh_) pkt_recvd = 0;
if (Pr < RXThresh_) pkt_recvd = 1; hdr->error() = 1;
else pkt_recvd = 1; hdr->error() = 0;
```

- mac/mac-802_11.cc

```
void Mac802_11::recv(Packet *p, Handler *h) {
…
if (rx_state_ == MAC_IDLE) {
...
} else {
        if(pktRx_->txinfo_.RxPr / p->txinfo_.RxPr >= p->txinfo_.CPThresh) capture(p);
        else collision(p);
}
```

If medium idle, receive it

If medium is not idle, receive it if SINR>CPThresh. Collision, otehrwise

c.yu91@csuohio.edu

Lab Exercise: Multirate Control

Data rate (Mbps)	Receive threshold (dBm)	Communication distance (m)	Capture threshold (dB)	Capture distance (m)
6	-82		6.02	
9	-81		7.78	
12	-79		9.03	
18	-77		10.79	
24	-74		17.04	
36	-70		18.80	
48	-66		24.05	
54	-65		24.56	

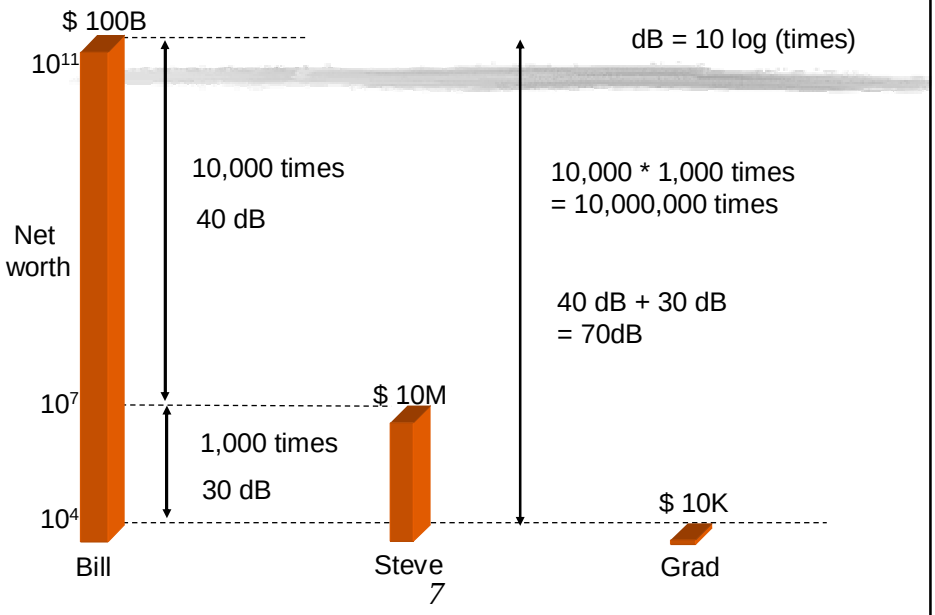
** Transmit power: 6 dBm*

** For a successful communication, when a node transmits a packet, other nodes should not send theirs. This can lead to capture distance.*

** CSThresh = -91 dBm*

c.yu91@csuohio.edu

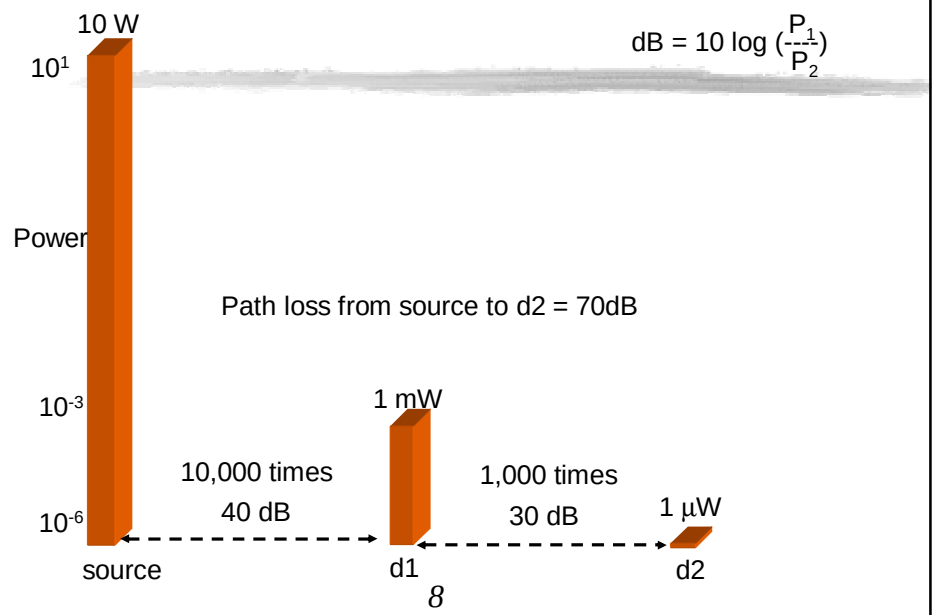
dB (relative measure)



7

c.yu91@csuohio.edu

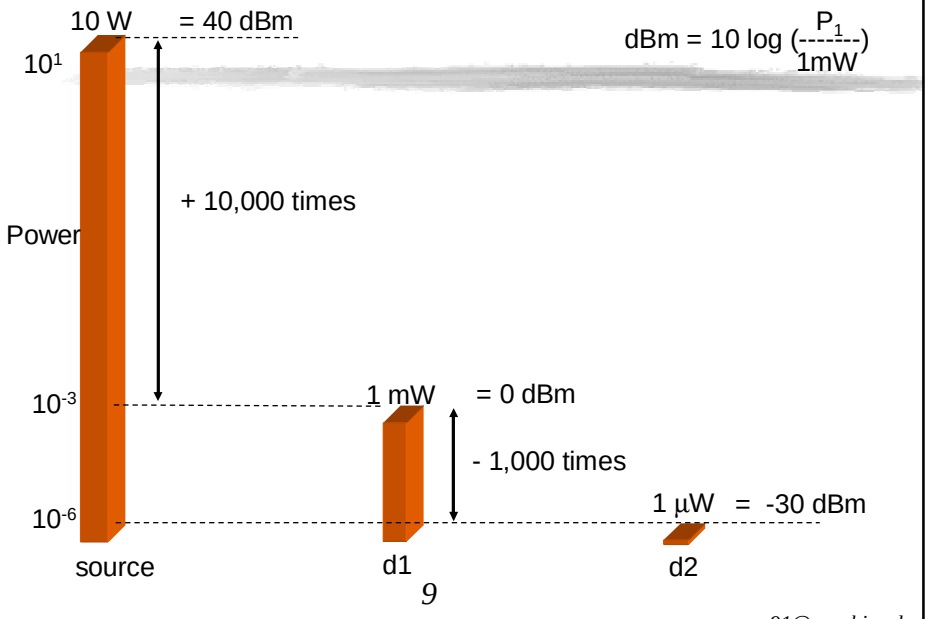
Path loss in dB



8

c.yu91@csuohio.edu

dBm (absolute measure of power)



9

c.yu91@csuohio.edu

18.1 Free space model

The free space propagation model assumes the ideal propagation condition that there is only one clear line-of-sight path between the transmitter and receiver. H. T. Friis presented the following equation to calculate the received signal power in free space at distance d from the transmitter [12].

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L} \tag{18.1}$$

where P_t is the transmitted signal power. G_t and G_r are the antenna gains of the transmitter and the receiver respectively. $L (L \geq 1)$ is the system loss, and λ is the wavelength. It is common to select $G_t = G_r = 1$ and $L = 1$ in *ns* simulations.

18.2 Two-ray ground reflection model

A single line-of-sight path between two mobile nodes is seldom the only means of propation. The two-ray ground reflection model considers both the direct path and a ground reflection path. It is shown [29] that this model gives more accurate prediction at a long distance than the free space model. The received power at distance d is predicted by

$$P_r(d) = \frac{P_t G_t G_r h_t^2 h_r^2}{d^4 L} \tag{18.2}$$

where h_t and h_r are the heights of the transmit and receive antennas respectively. Note that the original equation in [29] assumes $L = 1$. To be consistent with the free space model, L is added here.

10

c.yu91@csuohio.edu

Lab Exercise: Multirate Control

Data rate (Mbps)	Receive threshold (dBm)	Communication distance (m)	Capture threshold (dB)	Capture distance (m)	*Carrier sense distance (m)*
6	-82	238	6.02	337	*575*
9	-81	224	7.78	351	*576*
12	-79	200	9.03	336	*536*
18	-77	178	10.79	331	*509*
24	-74	150	17.04	400	*550*
36	-70	119	18.80	351	*470*
48	-66	95	24.05	389	*484*
54	-65	89	24.56	366	*455*

** Transmit power: 6 dBm*

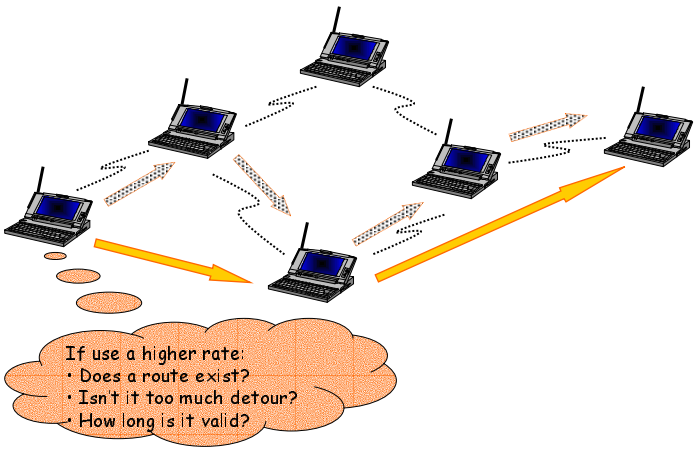
** For a successful communication, when a node transmits a packet, other nodes should not send theirs. This can lead to capture distance.*

** CSThresh = -91 dBm*

11

c.yu91@csuohio.edu

Lab Exercise: Multirate Control

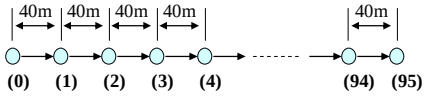


12

c.yu91@csuohio.edu

Lab Exercise: Multirate Control

- Goal: Compare the performance with different data rate (6, 18, 36 and 54 Mbps) with different mobility.
- Setup: 96 nodes in a chain. Nodes are separated by 40m. Traffic is node 1 to node 0 (CBR, 1000 bytes, 200 packets/second). Node 0 moves to the other end of the network with 0, 1, 5, 10, 20 m/second. Use AODV. (Need to change NETWORK_DIAMETER to 150 in aodv.h.) Use RTS/CTS.
- Output: Compare throughput



13

c.yu91@csuohio.edu

Lab Exercise: Multirate Control

- What does the data rate change affect?
  - Data rate (it will affect the packet time, etc.)
  - Receive threshold
  - Capture threshold
- How to make changes in ns-2?
  - Source file changes
  - Variable binding
  - Command method

14

c.yu91@csuohio.edu

Variable Binding

- Set/change a variable of C++ component from TCL run script
  - Advantage: No need to modify the source file (no need to compile & link)
- Special function “bind”
  - Interface: bind, bind_bw, bind_time, bind_bool

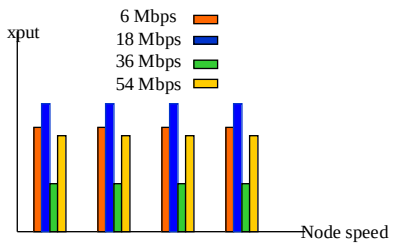
```
WirelessPhy::WirelessPhy() : Phy(), sleep_timer_(this), status_(IDLE) {
    bind("CPThresh_", &CPThresh_);
    bind("CSThresh_", &CSThresh_);
    bind("RXThresh_", &RXThresh_);
    bind("Pt_", &Pt_);
```

  - Default value is defined in $ns/tcl/lib/ns-default.tcl
    - Mac/802_11 set dataRate_ 1.0e6
    - Mac/802_11 set PLCPDataRate_ 1.0e6
    - Phy/WirelessPhy set CPThresh_ 10.0
    - Phy/WirelessPhy set RXThresh_ 3.652e-10
    - Phy/WirelessPhy set bandwidth_ 2e6
    - Phy/WirelessPhy set Pt_ 0.28183815

15

c.yu91@csuohio.edu

Lab Exercise: Multirate Control



16

c.yu91@csuohio.edu

Lab Exercise: Multirate Control

- Make your life a bit easier…
- Use a run batch file

```
TCL_FILE=carj.tcl
TRACE_FILE=out.tr
RES_AWK_FILE=delay.awk
RESULT_FILE=carj.out

for RATE in 06 18 36 54
  do
    for SPEED in 0 1 5 10 20
    do
      OPTION="$RATE $LAYOUT $SPEED"
      echo $OPTION ......
      echo $OPTION ...... >> $RESULT_FILE
      ns $TCL_FILE $OPTION
      awk -f $RES_AWK_FILE $TRACE_FILE >> $RESULT_FILE
    done
  done
```


c.yu91@csuohio.edu

Lab Exercise: Multirate Control

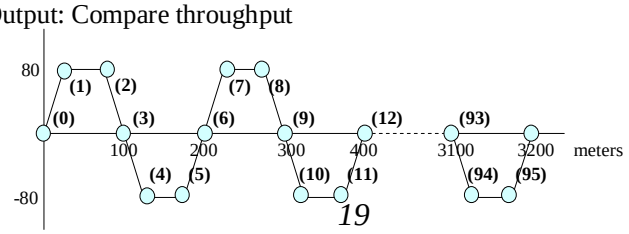
- How to pass values to TCL script?

```
set val(x)          4000

set val(y)          200
set val(nn)         96
set val(stop)       170

if {$argc != 3} {
      puts "Usage ns carj.tcl
      DataRate(06/18/36/54) Layout(chain/para)
      Speed(0/1/5/10/20)"
      exit
}

set par0 [lindex $argv 0]
set par1 [lindex $argv 1]
set par2 [lindex $argv 2]

if {$par0 == "06"} {
      Mac/802_11 set dataRate_       6.0e6
      Phy/WirelessPhy set bandwidth_ 6e6
      Mac/802_11 set PLCPDataRate_  6.0e6
      Phy/WirelessPhy set CPThresh_ ????
      Phy/WirelessPhy set RXThresh_ ????
}
…..
if {$par0 == "54"} {
      Mac/802_11 set dataRate_       54.0e6
      Phy/WirelessPhy set bandwidth_ 54e6
      Mac/802_11 set PLCPDataRate_  54.0e6
      Phy/WirelessPhy set CPThresh_ ????
      Phy/WirelessPhy set RXThresh_ ????
}
…..
…..
if {$par2 == "1"} {
      $ns_ at 0.0 "$node_(0) setdest 3400.0 1.0
      1.0"
}
…..
if {$par2 == "20"} {
      $ns_ at 0.000000000000 "$node_(0)
      setdest 3400.0 1.0 20.0"
}
```


c.yu91@csuohio.edu

Extra

- Goal: Compare the performance with different data rate (6, 18, 36 and 54 Mbps) with different mobility.
- Setup: 96 nodes in a quadrilateral topology. Traffic is node 1 to node 0 (CBR, 1000 bytes, 200 packets/second). Node 0 moves to the other end of the network with 0, 1, 5, 10, 20 m/second. Use AODV. (Need to change NETWORK_DIAMETER to 150 in aodv.h.) Use RTS/CTS.
- Output: Compare throughput



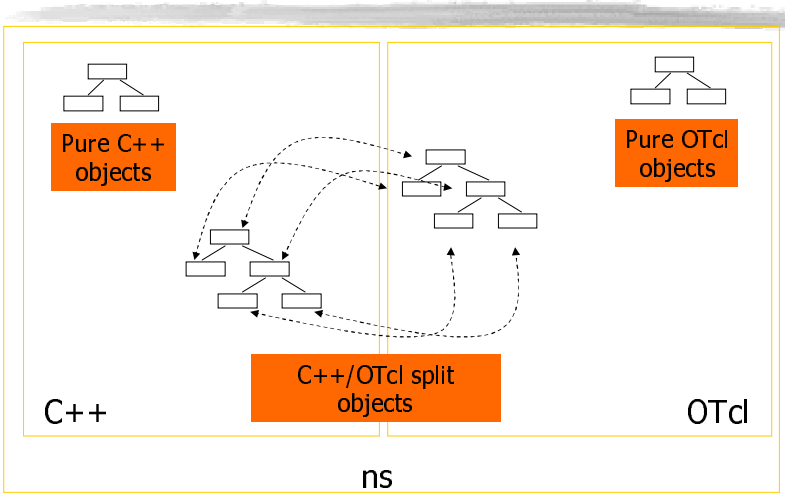
c.yu91@csuohio.edu

Access C++ component from TCL script via Command Method

- Command method to hand control to C++
  - Execute C++ function
  - Set/get C++ variables
  - Communicate with OTCL

c.yu91@csuohio.edu

OTcl and C++



21

c.yu91@csuohio.edu

NS2 Design: C++ and Otcl Separation

- C++ for data
  - Protocol implementation, such ac TCP.cc, UDP.cc
  - per packet action
- Otcl for control
  - Configuration and control in simulation
  - periodic or triggered action

\+ Compromize between composibility and speed

– Learning & debugging

22

c.yu91@csuohio.edu

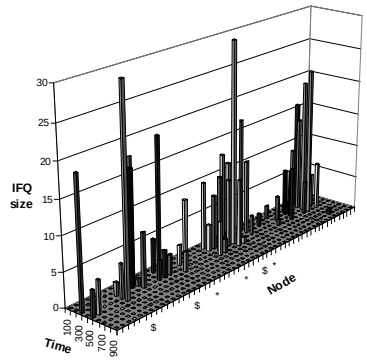
Homework #3

- Goal: Want to know that nodes undertake the role of packet forwarding uniformly – packet queue size distribution.
- Setup: 50 nodes in 300x1500 m^2 network based on random waypoint mobility with the maximum speed of 5m/s and the pause time is 0, 500 and 900 seconds. Simulation time is 900 seconds. 20 CBR sources send 4 512-byte packets per second. AODV is used. Use RTS/CTS.
- Output: Show IFQ sizes of all nodes at 100 second-interval. Throughput and delay curve. Explain what's happening.

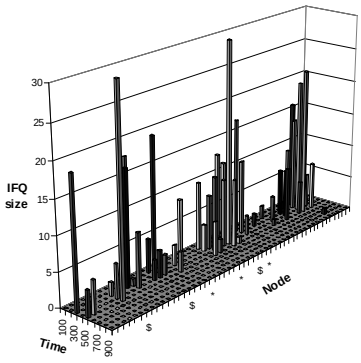
23

c.yu91@csuohio.edu

Homework #3



Pause = 0



Pause = 500

24

c.yu91@csuohio.edu

Command Method

- ~queue/priqueue.cc

```
int PriQueue::command(int argc, const char*const* argv)
{
  if (argc == 2 && strcasecmp(argv[1], "reset") == 0)
    {
      Terminate();
     //FALL-THROUGH to give parents a chance to reset
    }

/******************** ********************/
        static int              len = 0 ;
        if ( argc == 3   &&   strcmp( argv[1], "ifq-len" ) == 0 ) {
            len = length() ;
            FILE*               fp = fopen( argv[2], "a" ) ;
            fprintf( fp, "%d\t", len ) ;
            fclose( fp ) ;
            return (TCL_OK);
        }
        if ( argc == 3   &&   strcmp( argv[1], "ifq-len2" ) == 0 ) {
            FILE*               fp = fopen( argv[2], "a" ) ;
            fprintf( fp, "\n", len ) ;
            fclose( fp ) ;
            return (TCL_OK);
        }
/******************** ********************/

  return DropTail::command(argc, argv);
}
```

Don't forget to "Make"

c.yu91@csuohio.edu

Command Method

- **In the TCL script**

```
for {set i 0} {$i < $val(nn) } {incr i} {
        set node_($i) [$ns_ node]
        $node_($i) random-motion 0                ;# disable random motion
        set priQ_($i) [$node_($i) set ifq_(0)]
      }

$ns_ at 100.0  "getQLen $val(nn)"

proc getQLen { num } {
      global ns_ priQ_
      set interval 100.0
      for {set i 0} {$i < $num } {incr i} {
         $priQ_($i) ifq-len ifq-sz.tr
      }
      $priQ_(0) ifq-len2 ifq-sz.tr
       set now [$ns_ now]
       $ns_ at [expr $now+$interval] "getQLen $num"
}
```


c.yu91@csuohio.edu

Command Method

- ~tcl/ex/simple-wireless.tcl

```
for {set i 0} {$i < $val(nn) } {incr i} {
        set node_($i) [$ns_ node]
        $node_($i) random-motion 0          ;# disable random motion
        set mac_($i) [$node_($i) getMac 0]  ; added (see ~tcl/lib/ns-mobilenode.tcl)
}

for {set i 0} {$i < 150 } {incr i} {                ; added
      $ns_ at $i.01 "$mac_(0) new_command 0";  ; added
}                                                   ; added


% ns simple-wireless.tcl
```


c.yu91@csuohio.edu