ENSC 427 Communication Network
Spring 2012

Performance evaluation of TDMA Vs 802.11(CSMA)

Team 8:

Haishuo Zhang	301100089	hza43@sfu.ca
Changcheng Wang	301084351	ccw11@sfu.ca
Keren Wang	301097533	kwa35@sfu.ca

Abstract

Time division multiple access (TDMA) is a probabilistic media Access Control (MAC) protocol in which is a channel access method for shared medium network. It allows several users to share the same frequency channel by dividing the signal into different time slots[1]. Carrier Sense Multiple Access (CSMA) is also a probabilistic media Access Control (MAC) in which a node verifies the absence of other traffic before transmitting on a shared transmission medium [2].

In this project, we plan to use ns2 to simulate the TDMA to transmit data at first, and then simulate the data transfer with CSMA. After the simulations, we will use AWK to analysis data and then Xgraph to plot the end-to-end delay with two difference amount packet size 48 and 4800 for two protocols. After the simulation, TDMA have larger delay time than CSMA, but TDMA delay is more stable. Therefore, TDMA is a better protocol in our two testing cases.

Contents

1 Introduction:

Time division multiple access (TDMA), a second-generation (2G) technology used in digital cell telephone communication, is a probabilistic media Access Control (MAC) protocol in which is a channel access method for shared medium network [1]. It is a method that can divides the spectrum into time slots so that the more amounts of data can be carried [2]. In this way, specific frequency is not one-to -one to a user, but a single frequency can provide multiple data channels to co response multiple users. Although TDMA is considered the most inadvanecd second-generation technology, it was widely used all over the world in the past few years. The statistics shows that about 9% digital cell phone user chose TDMA in US in 1999[2].Like TDMA, CSMA (Carrier Sense Multiple Access) is also a probabilistic media Access Control (MAC). It can detect the absence of other traffic before transmitting on a shared transmission medium [3]. There are two modifications of CSMA, the one is called CSMA/CD, and the other one is CSMA/CA which is used in our project, where CD is short for collision detection and CA stand for collision avoidance. CSMA/CD refers to use the terminating transmission to detect and deal with the collision and then prevent the collision happening again to improve the performance. On the other hand, CSMA/CA, it acts to reduce the probability of the first-time collision happen on the channel. Because it checks if the channel is clear or not once a node receives a packet. If the channel is idle, then the packet is sent; however, if the channel is busy, the node will wait a period of time and then check it again until the channel is clear.

That ensure there is only one node is transmitting at one time so that prevent the collision fundamentally [4].In this project, we plan to use network simulator (ns2) to simulate the data transfer with TDMA protocol and 802.11 (CSMA\CA) protocol. According to the end-to-end delay, we will evaluate their performance with different amount of packet size. During our simulation, we decide two setting points of packet size which are 48 and 4800. In the end, we are going to compare and analysis the result.

2 Main Sections:

2.1 Description of the entire design

In our project, we use "Network Simulator-2" to simulation the data transfer in TDMA protocol and CSMA protocol, and compare their performances. At first, we write some codes about TDMA protocol [appendix-tdma.tcl] and CSMA protocol [appendix-csma.tcl] for NS2, and run the simulation. After the simulation is done, using the *delay.awk* [appendix-delay.awk] to filter out all useful data and save into *TDMA* or *CSMA* file. At last, we use Xgraph to plot the end-to-end delay of two difference protocols in the same graph. We will run the same simulation twice, but they have the difference packet sizes which are 48 and 4800 (Figure1)

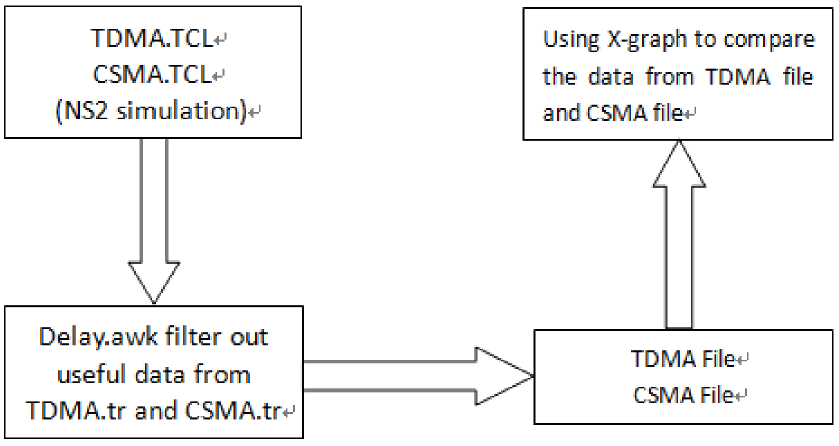
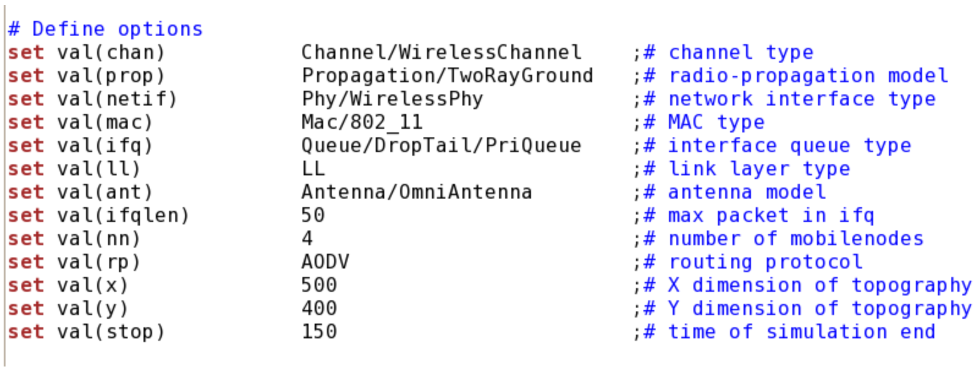

TDMA.TCL
CSMA.TCL
(NS2 simulation)
Delay.awk filter out useful data from TDMA.tr and CSMA.tr
TDMA File
CSMA File
Using X-graph to compare the data from TDMA file and CSMA file

Figure1. Flow chart of the project

2.2 TDMA and CSMA

a) Set some initial options such as channel type, network interface type and so on. The values which are from *val(chan)* to *val(nn)* are for nodes. *Val(x),Val(y),* and *Val(stop)* are for ns2 (Figure2)

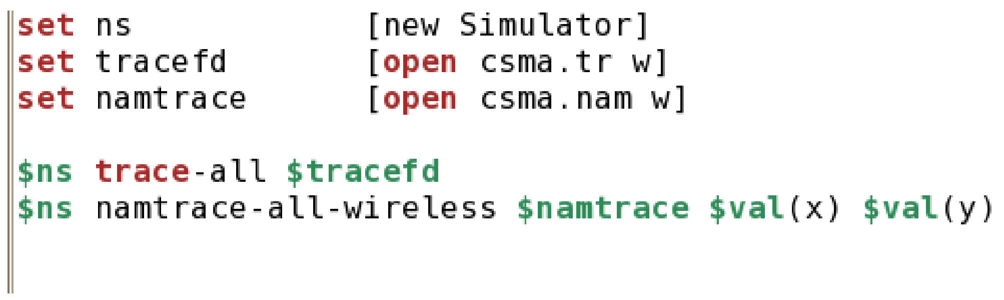
Note: Mac/802_11 is CSMA; Mac/Tdma is TDMA



```
# Define options
set val(chan)           Channel/WirelessChannel    ;# channel type
set val(prop)           Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)          Phy/WirelessPhy            ;# network interface type
set val(mac)            Mac/802_11                 ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)             LL                         ;# link layer type
set val(ant)            Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)         50                         ;# max packet in ifq
set val(nn)             4                          ;# number of mobilenodes
set val(rp)             AODV                       ;# routing protocol
set val(x)              500                        ;# X dimension of topography
set val(y)              400                        ;# Y dimension of topography
set val(stop)           150                        ;# time of simulation end
```

Figure2. Define the options

b) Set up a new Simulator; create *TDMA.tr* and *TDMA.nam* or *CSMA.tr* and *CSMA.nam* in order to record the data during the simulation (Figure3)

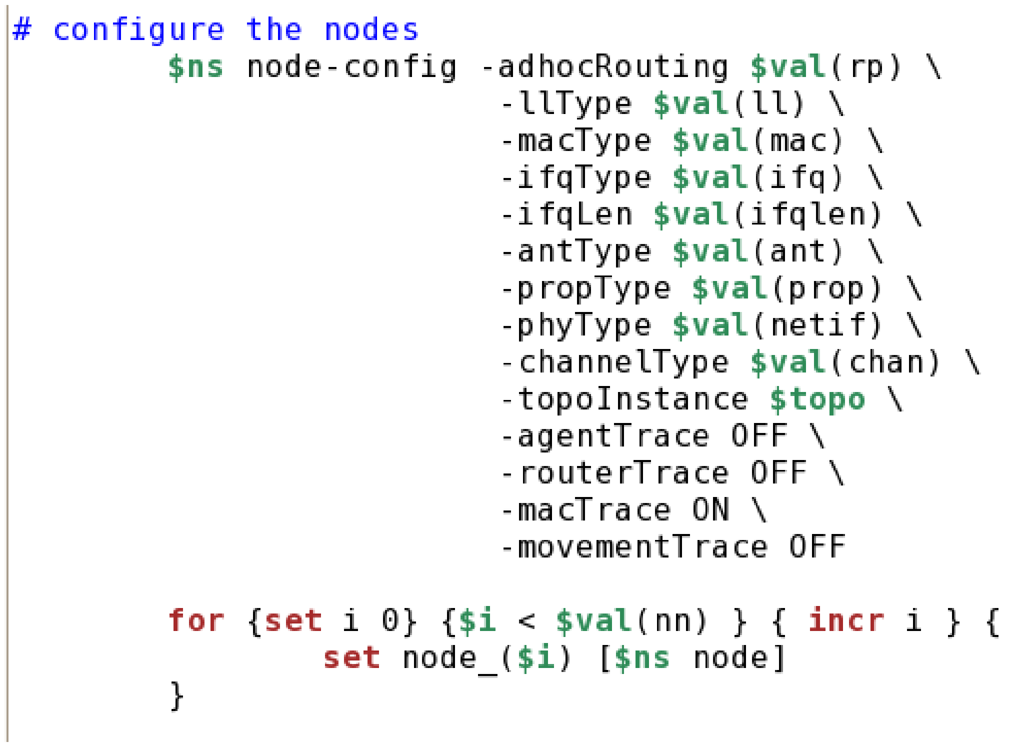


```
set ns              [new Simulator]
set tracefd         [open csma.tr w]
set namtrace        [open csma.nam w]

$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)
```

Figure3. Set the simulator and output files

c) Build topography, and sign a *God* which can manage nodes and make sure all nodes are in the area.

d) Configure the nodes, but only turn on *macTrace*; also create 4 nodes(Figure4)

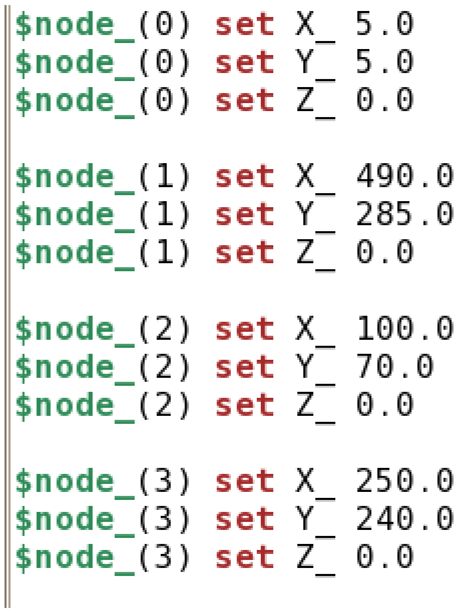


```
# configure the nodes
        $ns node-config -adhocRouting $val(rp) \
                         -llType $val(ll) \
                         -macType $val(mac) \
                         -ifqType $val(ifq) \
                         -ifqLen $val(ifqlen) \
                         -antType $val(ant) \
                         -propType $val(prop) \
                         -phyType $val(netif) \
                         -channelType $val(chan) \
                         -topoInstance $topo \
                         -agentTrace OFF \
                         -routerTrace OFF \
                         -macTrace ON \
                         -movementTrace OFF

        for {set i 0} {$i < $val(nn) } { incr i } {
                set node_($i) [$ns node]
        }
```

Figure4. Configure the nodes and create nodes

e) Provide the initial location of nodes, so that it is easier to monitor. (Figure5)

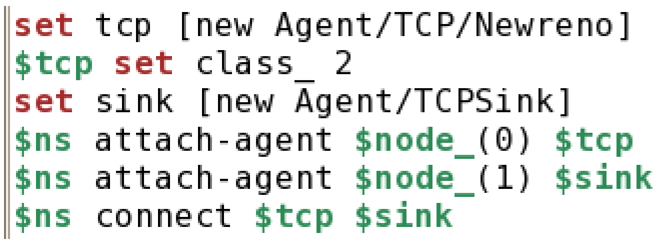
f) Assign the node0 is a sender, the node1 is a receiver, and node2,3 are transporters, where X,Y,Z value represents the coordinate. Since it is a 2D graph, all the Z values are 0. For example, node 0 is located at 5.0,5.0



```
$node_(0) set X_ 5.0
$node_(0) set Y_ 5.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 490.0
$node_(1) set Y_ 285.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 100.0
$node_(2) set Y_ 70.0
$node_(2) set Z_ 0.0

$node_(3) set X_ 250.0
$node_(3) set Y_ 240.0
$node_(3) set Z_ 0.0
```

Figure5. Locating the nodes

g) Set a TCP connection between node0 and node1(Figure6)



```
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(0) $tcp
$ns attach-agent $node_(1) $sink
$ns connect $tcp $sink
```

Figure6. Set TCP connection between Node0 and Node1

h) Attach the CBR into the TCP, define some parameter; CBR start sending data every second after 10s later (Figure7)

```
set e [new Application/Traffic/CBR]
$e attach-agent $tcp
$e set packetSize_ 4800
$e set rate_ 6kb
$e set interval_ 1
$ns at 10.0 "$e start"
```

Figure7. Creating the CBR and attach into the TCP

i) Reset the nodes and stop the simulation at 150s, and generate *.tr* file and *.nam* file

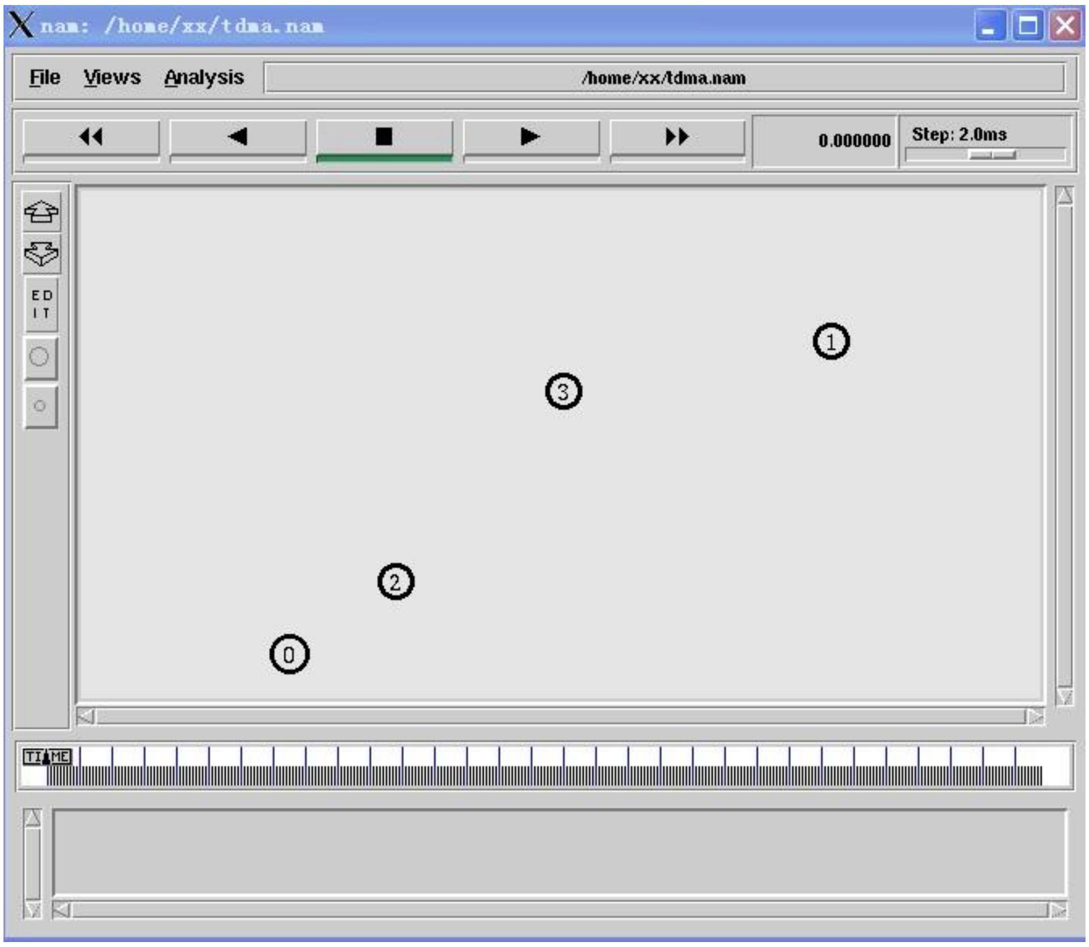


Figure8. nam scenario

2.3 AWK

The AWK utility is a data extracting and reporting tool that lets programmers use various forms of statements to write programs that consists of patterns, which are to be searched for in each line of the document and the program's action that will take place when a match is found.[5]

*"**AWK** is a language for processing text files. A file is treated as a sequence of records, and by default each line is a record. Each line is broken up into a sequence of fields, so we can think of the first word in a line as the first field, the second word as the second field, and so on. An AWK program is of a sequence of pattern-action statements. AWK reads the input a line at a time. A line is scanned for each pattern in the program, and for each pattern that matches, the associated action is executed." - Alfred V. Aho[6]*

The main purpose of AWK filters out the useful data from *TDMA.tr* or *CSMA.tr*. In *TDMA.tcl* and *CSMA.tcl*, we record all of data in the Mac layer, and store them into *TDMA.tr* and *CSMA.tr*; however, we are only interest in when node0 send the packet to node1, and when node1 receive the same packet. Once we collect the time about sending and receiving. We just subtract the sending time from the receiving time in order to get the delay time for one signal packet. If we get all the delay time for every transported packets, then we save these data into *TDMA* and *CSMA*

2.4 **Xgraph**

After the AWK filters out the useful data and store into 2 files, we just use Xgraph software to plot the end-to-end graph. Figure9 and Figure10 are the results, where X axis is the simulating time; Y axis is the delay time and red curve represents TDMA and green curve stands for CSMA. From these two figures we can see, End-to-End delay of TDMA is much larger than CSMA's with two packet sizes. The delay of TDMA with packet size 48 is about 82.5*10^{-3} second. Whereas the CSMA is about 30*10^{-3} second; however, for the larger packet size 4800, CSMA ETE delay is really unstable and TDMA delay is stably keep at the level of 142*10^{-3} second after a significant bomb from the beginning of simulating time.

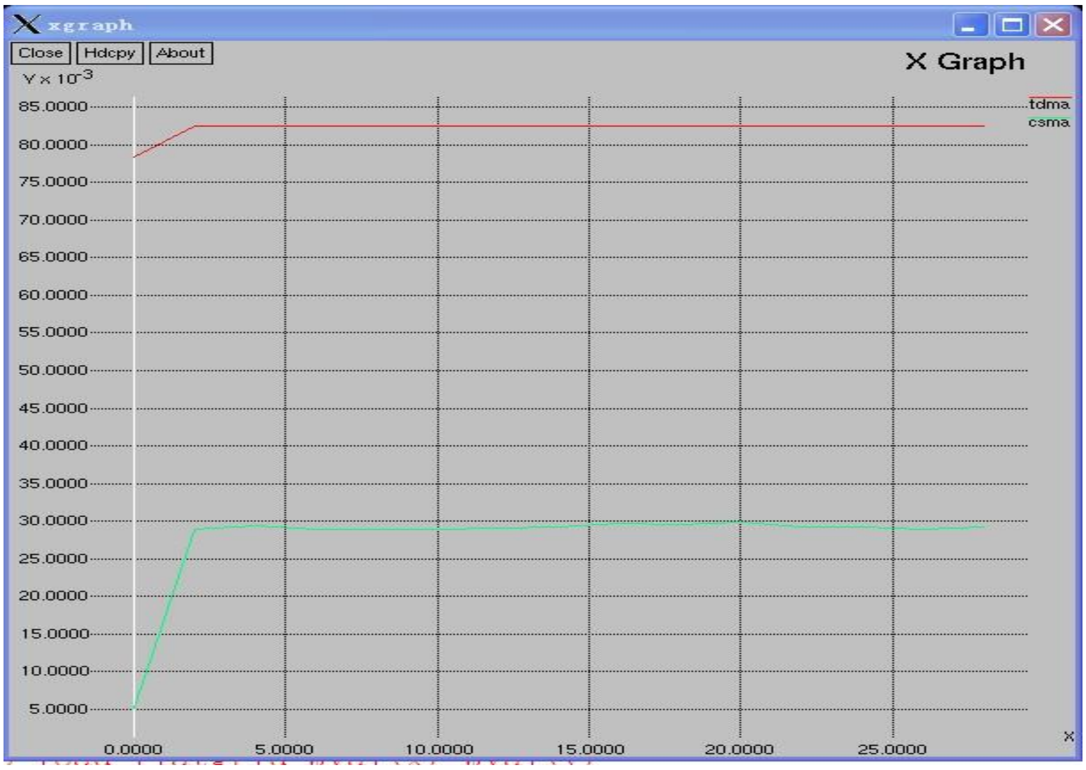


Figure9 End-to-End delay with packet size 48

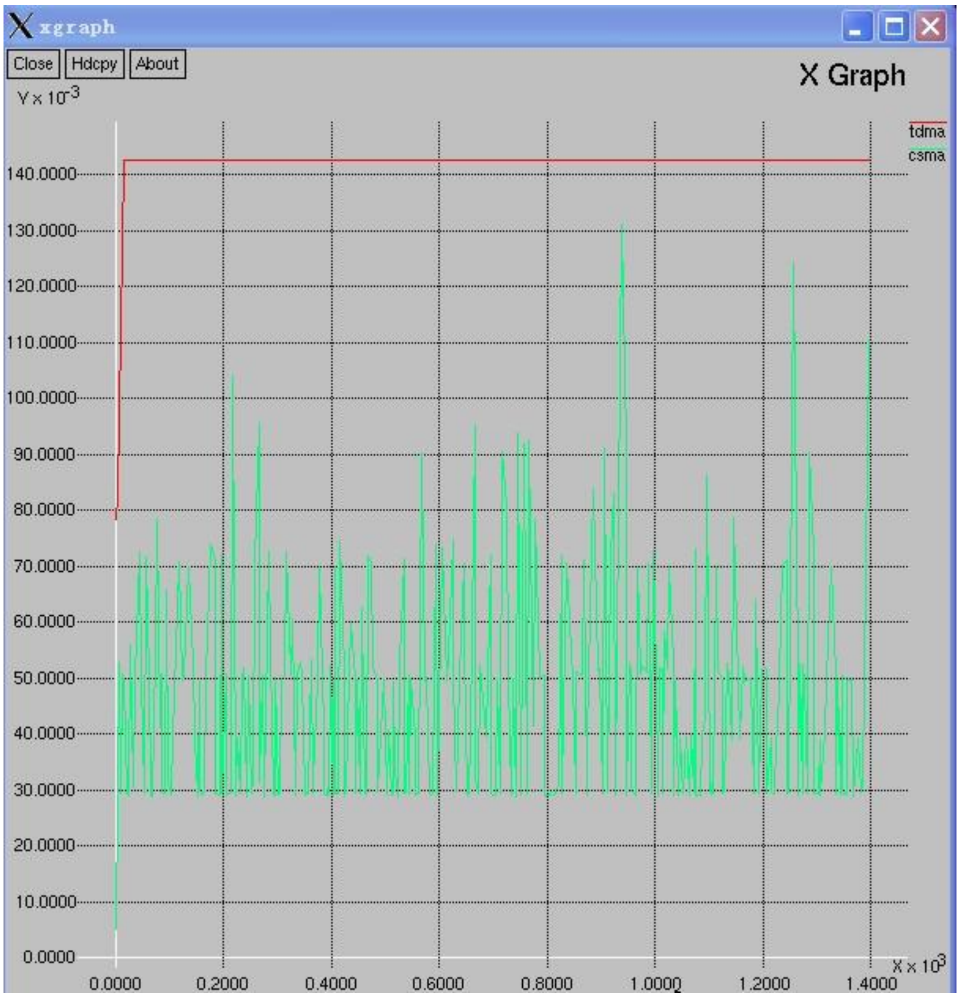


Figure10 End-to-End delay with packet size 4800

3 Discussion and Conclusion:

3.1 Conclusion

From the simulation result, we found that if network data volume is within the network can withstand, TDMA will waste a lot of slot time. On the other hand, in the same situation, the end to end delay of CSMA is smaller than that of TDMA. Comparing figure9 and figure10, we can find that with a lager throughput, the delay of TDMA network transmission tends to stably, but the delay of CSMA transmission is getting lager and unstable. Even though from the graphs, the delay of CSMA is still smaller than the delay of TDMA, we can expect easily that with the increasing of the throughput, the delay of CSMA will be large than the delay of TDMA and because of the instability of CSMA, it may be disconnected in a very large throughput.

Today, huge amount of data transfer is required. Even though CSMA has a smaller delay, TDMA is much more stable. Therefore, we conclude that TDMA is a better protocol in our two testing cases.

3.2 Difficulty

In this simulation, we have to write three tcl codes for CSMA, TDMA and Xgraph. Since we cannot use the .tr file to get the xgraph directly, we have to write our simulation result to another file. Then we use the data to get the graph we want.

3.3 Future Work

In this project, only simulate two packet sizes. To make the simulation more accurate, more packets sizes can be simulated and a new graph, delay vs. size of packets graph, can be drawn. To make a better protocol, TDMA and CSMA can be combined to a new protocol. In small amount of data, the new protocol performs as CSMA and in huge amount of data, the new protocol performs as TDMA

4 Reference:

1. Ian F. Akyildiz and Janise McNair, Medium Access Control Protocols for Multimedia Traffic in Wireless Networks[A], Georgia Institute of Technology Loren Carrasco Martorell and Ramon Puigjaner, Universitat de les llles Balears Yelena Yesha, University of Maryland at Baltimore Count, July/August. 2009.

2. TDMA IS-136 (Time Division Multiple Access).[Online]
http://www.mobilecomms-technology.com/projects/tdma_is136/

3. Carries Sense Multiple Access[Online]
http://en.wikipedia.org/wiki/Carrier_sense_multiple_access#References

4. Nat. Inst. of Inf. & Commun. Technol., Yokosuka, Japan
Harada, H. ; Kato, S. Nat. Inst. of Inf. & Commun. Technol., Yokosuka, Japan[A], 13-16 Sept. 2009

5. Jose Nazario, An Introduction to AWK[A]. Linux Journal, Mar 08, 2006.

6. The A-Z of Programming Languages: AWK
http://www.computerworld.com.au/index.php/id;1726534212;

7. Leonidas Georgiadis, Carrier-Sence Multiprle Access (CSMA) Protocols[A], February 13, 2002

8. Jean Torrilhes, The MAC level (link layer), 3 August 2000 [online]

http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Linux.Wireless.mac.html4.

9. Marek Miśkowicz, Analysis of Mean Access Delay in Variable-Window CSMA[A], AGH University of Science and Technology, Department of Electronics, al. Mickiewicza 30,

10.Simon S. Lam, Delay Analysis of Packet-switched TDMA System[A], IBM Thomas J Watson Research Center Yorktown Heights, New York 10598

5 Appendix

5.1 Code Listing

5.1.1 tdma.tcl

```
# Define options
set val(chan)          Channel/WirelessChannel     ;# channel type
set val(prop)          Propagation/TwoRayGround    ;# radio-propagation model
set val(netif)         Phy/WirelessPhy             ;# network interface type
set val(mac)           Mac/Tdma                    ;# MAC type
set val(ifq)           Queue/DropTail/PriQueue     ;# interface queue type
set val(ll)            LL                          ;# link layer type
set val(ant)           Antenna/OmniAntenna         ;# antenna model
set val(ifqlen)        50                          ;# max packet in ifq
set val(nn)            4                           ;# number of mobilenodes
set val(rp)            AODV                        ;# routing protocol
set val(x)             500                ;# X dimension of topography
set val(y)             400                ;# Y dimension of topography
set val(stop)          150                      ;# time of simulation end

set ns          [new Simulator]
set tracefd        [open tdma.tr w]
set namtrace         [open tdma.nam w]

$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)


# set up topography object
set topo         [new Topography]

$topo load_flatgrid $val(x) $val(y)

create-god $val(nn)

#
#   Create nn mobilenodes [$val(nn)] and attach them to the channel.
#

# configure the nodes
        $ns node-config -adhocRouting $val(rp) \
             -llType $val(ll) \
             -macType $val(mac) \
                    -ifqType $val(ifq) \
                    -ifqLen $val(ifqlen) \
                    -antType $val(ant) \
                    -propType $val(prop) \
                    -phyType $val(netif) \
                    -channelType $val(chan) \
                    -topoInstance $topo \
                    -agentTrace OFF \
                    -routerTrace OFF \
                    -macTrace ON \
                    -movementTrace OFF

    for {set i 0} {$i < $val(nn) } { incr i } {
        set node_($i) [$ns node]
    }

# Provide initial location of mobilenodes
$node_(0) set X_ 5.0
$node_(0) set Y_ 5.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 490.0
$node_(1) set Y_ 285.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 100.0
$node_(2) set Y_ 70.0
$node_(2) set Z_ 0.0

$node_(3) set X_ 250.0
$node_(3) set Y_ 240.0
$node_(3) set Z_ 0.0

# Set a TCP connection between node_(0) and node_(1)
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(0) $tcp
$ns attach-agent $node_(1) $sink
$ns connect $tcp $sink
set e [new Application/Traffic/CBR]
$e attach-agent $tcp
$e set packetSize_ 4800
$e set rate_ 6kb
```

```
$e set interval_ 1
$ns at 10.0 "$e start"


# Define node initial position in nam
for {set i 0} {$i < $val(nn)} { incr i } {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}

# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
        $ns at $val(stop) "$node_($i) reset";
}

# ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 150.01 "puts \"end simulation\" ; $ns halt"
proc stop {} {
        global ns tracefd namtrace
        $ns flush-trace
        close $tracefd
        close $namtrace
        #Execute nam on the trace file
        exec nam tdma.nam &
        exit 0
}

#Call the finish procedure after 5 seconds of simulation time
$ns run

```

5.1.2 csma.tcl

```
# Define options
set val(chan)           Channel/WirelessChannel    ;# channel type
set val(prop)           Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)          Phy/WirelessPhy            ;# network interface type
set val(mac)            Mac/802_11                 ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)             LL                         ;# link layer type
set val(ant)            Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)          50                          ;# max packet in ifq
set val(nn)               4                          ;# number of mobilenodes
set val(rp)               AODV                       ;# routing protocol
set val(x)                500                    ;# X dimension of topography
set val(y)                400                ;# Y dimension of topography
set val(stop)        150             ;# time of simulation end

set ns             [new Simulator]
set tracefd           [new open csma.tr w]
set namtrace          [open csma.nam w]

$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)


# set up topography object
set topo           [new Topography]

$topo load_flatgrid $val(x) $val(y)

create-god $val(nn)

#
#   Create nn mobilenodes [$val(nn)] and attach them to the channel.
#

# configure the nodes
        $ns node-config -adhocRouting $val(rp) \
             -llType $val(ll) \
             -macType $val(mac) \
             -ifqType $val(ifq) \
             -ifqLen $val(ifqlen) \
             -antType $val(ant) \
             -propType $val(prop) \
             -phyType $val(netif) \
             -channelType $val(chan) \
             -topoInstance $topo \
             -agentTrace OFF \
             -routerTrace OFF \
             -macTrace ON \
             -movementTrace OFF

    for {set i 0} {$i < $val(nn) } { incr i } {
        set node_($i) [$ns node]
        }

# Provide initial location of mobilenodes
$node_(0) set X_ 5.0
$node_(0) set Y_ 5.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 490.0
$node_(1) set Y_ 285.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 100.0
$node_(2) set Y_ 70.0
$node_(2) set Z_ 0.0

$node_(3) set X_ 250.0
$node_(3) set Y_ 240.0
$node_(3) set Z_ 0.0

# Set a TCP connection between node_(0) and node_(1)
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(0) $tcp
$ns attach-agent $node_(1) $sink
$ns connect $tcp $sink
set e [new Application/Traffic/CBR]
$e attach-agent $tcp
$e set packetSize_ 4800
$e set rate_ 6kb
$e set interval_ 1
$ns at 10.0 "$e start"

# Define node initial position in nam
for {set i 0} {$i < $val(nn)} { incr i } {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}

# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
        $ns at $val(stop) "$node_($i) reset";
}

```

```
# ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 150.01 "puts \"end simulation\" ; $ns halt"
proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
    #Execute nam on the trace file
      exec nam csma.nam &
    exit 0
}

#Call the finish procedure after 5 seconds of simulation time
$ns run
```

5.1.3 delay.awk

```
#Initial the program, and set max packetID# to 0
BEGIN {
        highest_uid = 0;

}

#Define every column of the tr file

{

        event = $1; #first column means that actions such as receiving and sendin.

        time = $2;          #time

        node_nb = $3;           #node number

        node_nb=substr(node_nb,2,1);      #in tr file, node number is like _3_, but we only need 3, so we remove '_'

        trace_type = $4;       #trac_type such as Mac

        flag = $5;                   #

        uid = $6;          #packetID

        pkt_type = $7;              #Packet type

        pkt_size = $8;       #packet size

    # Record the max CBR packet ID in Mac, and assign this ID to highest _uid

        if ( event=="s" && node_nb==0 && pkt_type=="cbr" && uid > highest_uid && trace_type=="MAC")
        {

                highest_uid = uid;
        }

        #record the packet sending time and save into start_time array

        if ( event=="s" && node_nb==0 && pkt_type=="cbr" && uid==highest_uid&&trace_type=="MAC" && pkt_size="112")
{
                start_time[uid] = time;
}
     #record the packet receiving time and save into end_time array

        if ( event=="r" && node_nb ==1 && pkt_type=="cbr" && uid==highest_uid&& trace_type=="MAC" && pkt_size="112")

                end_time[uid] = time;

}

END {

        # calculate the delay time

        for ( packet_id = 0; packet_id <= highest_uid; packet_id++ )

        {
                start = start_time[packet_id];
                end = end_time[packet_id];
                packet_duration = end - start;

             #print out the positive delay time
               if ( start <end )
               printf("%d %f\n", packet_id, packet_duration);
```

```
            }
}
```

5.1.4 Result.tcl

```
set ns              [new Simulator]
$ns at 1.0 stop
$ns at 1.0 "puts \"print xgraph\" ; $ns halt"
proc stop {} {
        # we save the useful data in tdma and csma file, xgraph plot them in a graph
      exec xgraph tdma csma -geometry 800x400 &

        exit 0
}
#Call the finish procedure after 5 seconds of simulation time
$ns run
```