DCU Engineering & Computing
Assignment Submission

Student Name(s):	Fei Wang
Student Number(s):	59211876
Programme:	MTCC - MEng in Telecommunications Engineering
Project Title:	EE500 Assignment
Module code:	EE500
Lecturer:	Gabriel-Miro Muntean
Project Due Date:	29-APR-2011

Signed: <u>Fei Wang</u>

Contents

1 Part A – Static Model

1.1 Model and Parameter

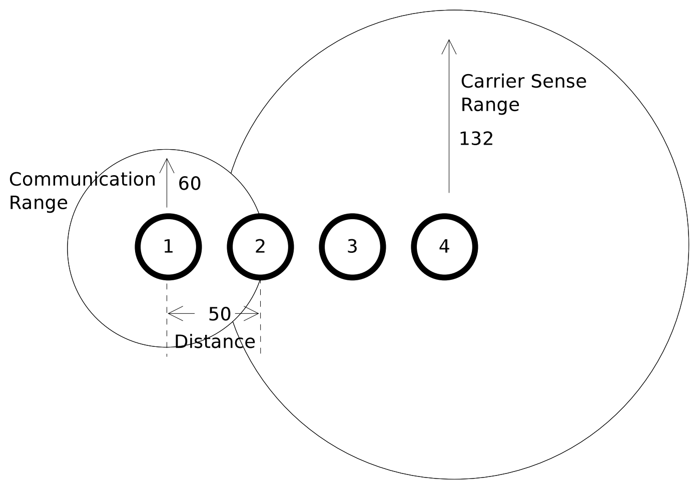

Figure 1.1: Static Traffic Model

As figure 1.1 show, this is our wireless hoc network topo. And the key parameter of model show in table 1.1:

Value / Parameter	Value
CSThresh_	5.00522e-12
RXThresh_	2.42253e-11
Pt_	0.000935073
freq_	2.472e9

Table 1.1: Key Parameter of model

How to determine these parameter. We know `RXThresh_` is the communication range between two nodes. `CSThresh_` is the carrier sense range. These two parameter is key important factor that affect communication. In TwoRayGround model. `Pr` is:

$$Pr = \frac{Pt * Gt * Gr * \lambda^2}{(4 * pi * d)^2 * L}$$

and d is the communication range. The rest parameter will keep the default value as them in ns-2 except Pt. Pt is the transmit power and interrelated with d as:

$$Pt = 7.21505664 * d^4 * e^{-11}$$

when $d = 250$, $Pt = 0.28183815$. In our model, $d = 60$, so $Pt = 0.000935073$. So we can determine the CSThresh_ and RXThresh_ with threshold.cc tool. Just like:

```
./threshold -Pt 0.000935073
            -fr 2.472e9
            -m TwoRayGround 60
```

RXThresh_ is assigned to this command ouput value, that is $2.42253e^{-11}$. In ns-2, Carrier sense range is communication range 2.2 times. So the carrier range is 132. Hence:

```
./threshold -Pt 0.000935073
            -fr 2.472e9
            -m TwoRayGround 132
```

And $CSThresh\_ = 5.00522e^{-12}$.

Parameter \ Value	Value
CWMin_	31
CWMax_	1023
SlotTime_	0.000020
SIFS_	0.000010
PreambleLength_	144
PLCPHeaderLength_	48
PLCPDataRate_	11.0e6
dataRate_	11.0e6
basicRate_	1.0e6

Table 1.2: 802.11b Standard Parameter

The table 1.2 show the key parameter of 802.11b standard. In this table, I just list the parameter that compare with 802.11 is different.

1.2 Performance Analysis

1.2.1 Delay Compare

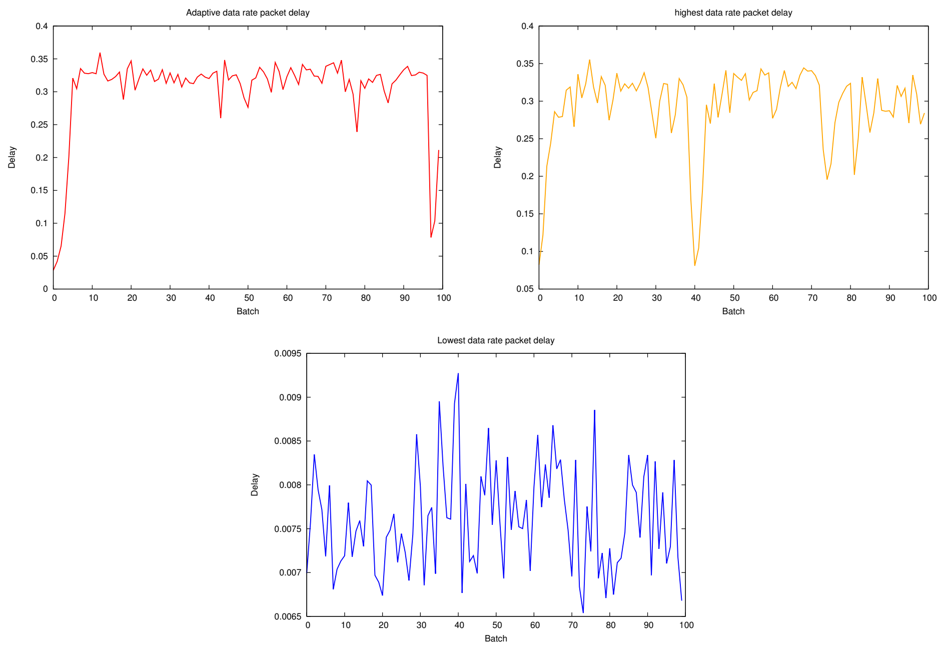

Figure 1.2: Delay Compare

	Avg[1]	Stdev[2]
AD	0.285798	0.073276
HI	0.302772	0.058543
LO	0.007705	0.003792

Table 1.3: Delay Compare

Because of the total packet is too much so that it is not a good idea to plot every packet end-to-end to the figure. I divide the packet to one hundred batch by adjusting the number of packet of each batch. Not only make the figure clear. But also can describe the trends of vary accurately. This approach also used in the remaining figure.

In the top-left, that is adaptive rate case. The highest rate case resides on the top-right. The third figure is the lowest rate case. This location arrangement will keep consistent in entire report.

[1]Avg is the abbreviation of the Average
[2]Stdev is the abbreviation of the Standard deviation

As figure 1.2 and table 1.3 illustrated. Highest rate leading the highest delay and the delay of lowest rate is the minimum. Adaptive rate between them. Note that the `Stdev` tell us that the jitter of delay. Since adjust delivery rate based on the loss ratio. It result in the maximum delay jitter.

1.2.2 Delay Jitter Compare

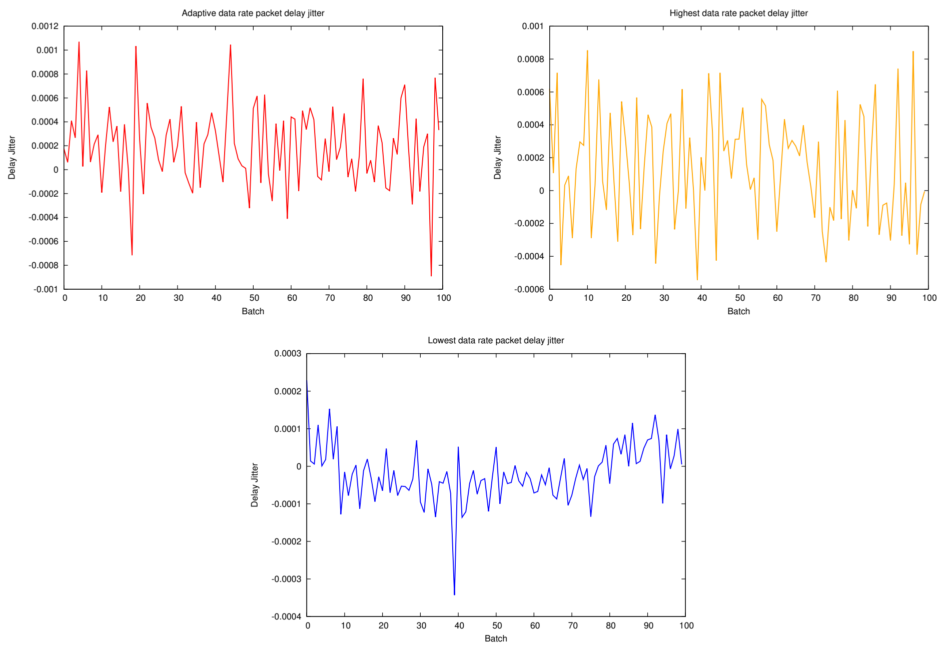

Figure 1.3: Delay jitter Compare

Define `jitter` as:

$$jitter = \frac{delay_2 - delay_1}{pkt\_id_2 - pkt\_id_1}$$

Intuitively, Highest rate case jitter mostly above zero. Lowest rate case jitter mostly below zero. By contrast, Adaptive rate case jitter around zero. This explain why the standard deviation of delay of adaptive rate case is maximum.

Why the delay of highest rate case is the maximum value among three case? As we know, end-to-end delay `T` is equal to:

$$T = W + S$$

`W` is queuing time and `S` is transmission time. Because of the other affective factor is same, We can simple assume the transmission time is same to three case. Hence we just need to think of how the delivery rate influence the queuing time `W`. Basically, the higher deliverate, the easier it is to casuse congestion and the higher queuing time. So it explain why `HI` rate case with highest delay time and mostly delay jitter above zero.

1.2.3 Throughput Compare

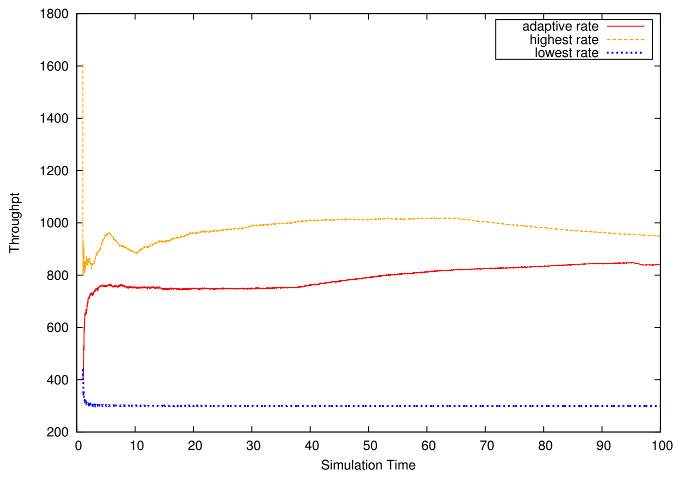

Figure 1.4: Throughput Compare

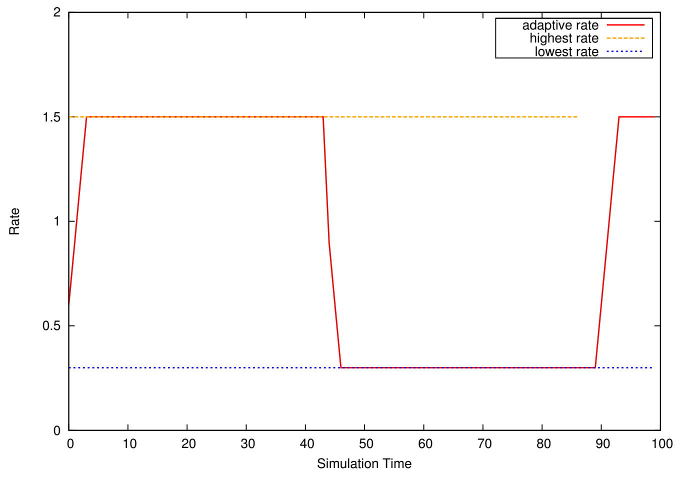

Figure 1.5: Adaptive Rate During Simulation Time

	Avg	Stdev
AD	811.203610	21.317470
HI	996.501545	39.579208
LO	306.456949	6.583978

Table 1.4: Throughput Compare

We can find the throughput of AD case keep a stable value even thought the delivery rate adjust with the loss ratio. But also the standard deviation of throughput of AD case is smaller than HI case. The throughput curve of AD case is not far away from HI case.

The figure also indicate that the three case get the highest throughput in the initial stage. After this stage, the throughput will keep approximate constant. That is said that network has a capacity.

We can simply conclude that the higher delivery rate, we can get the higher throughput from this figure. However, network has a capacity. So if the delivery rate reach the congestion collapse. This conclusion would be not correct.

1.2.4 Loss Rate Compare

	Send Packets	Receive Packets	Loss Rate
AD	10682	9926	0.070773
HI	12825	11675	0.089669
LO	4127	3709	0.101284

Table 1.5: Loss Compare

Because, this streams multimedia application adapts the delivery rate based on loss ratio. So the AD case get the lowest loss ratio. But counterintuitive, we found that the lowest delivery rate reach the highest loss ratio. The reason for this problem may be the `CBR` application between `node_1` and `node_4`. This QOAS application content the resource with CBR. Because of the lowest delivery rate, it will fail almost time. So the loss ratio of `LO` case is maximum.

1.2.5 PSNR Compare

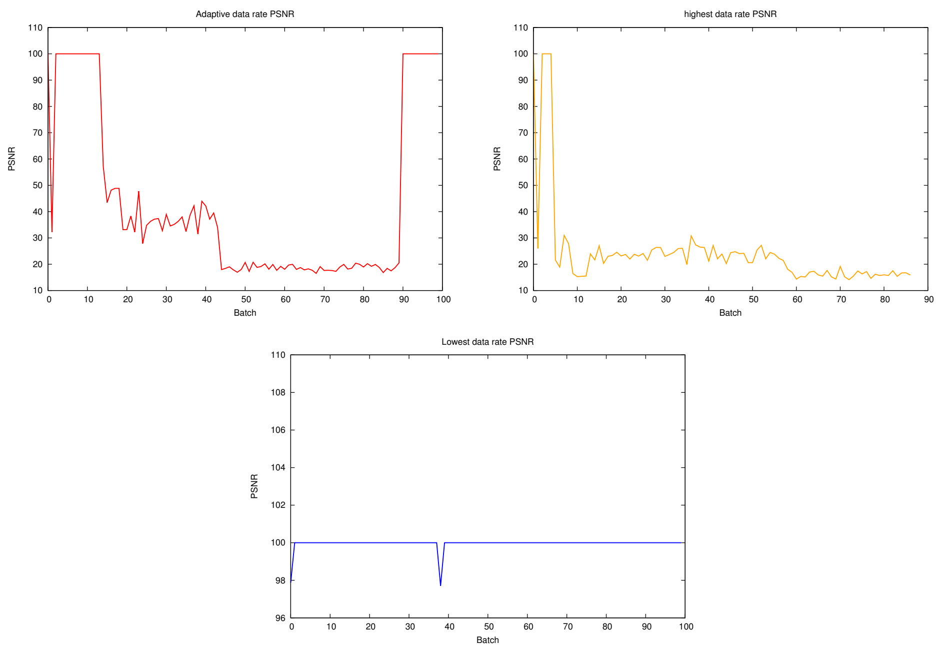

Figure 1.6: PSNR Compare

	Avg	Stdev
AD	49.662991	30.957876
HI	24.298941	17.519222
LO	99.901597	3.353683

Table 1.6: PSNR Compare

PSNR is estimated according to the formula present below:

$$PSNR \;=\; 20 \cdot log_{10}\left(\frac{MAX\_Bitrate}{\sqrt{(EXP\_Thr - CRT\_Thr)^2}}\right)$$

Basically, The higher PSNR, the communication quality reach higher. We can found that the PSNR of AD jitter sharply. The highest PSNR archive when the delivery rate is lowest. But PSNR of the AD case should be enough.

1.2.6 Summary

Performance Case	$\overline{Delay}$	$\overline{Throughput}$	$\overline{Loss}$	$\overline{PSNR}$
AD	②	②	①	②
HI	③	①	②	③
LO	①	③	③	①

Table 1.7: Best Performance Sorting

To sum up, I give a rank table between three case. ① mean the best choice in corresponding performance. ③ mean the worst choice. Comprehensive consideration of various factors, I think the `QOAS_AD` is the best solution among three case.

2 Part B – Movable Model

2.1 Performance Analysis

In order to give enough chance to communicate between `node_1` and `node_4`. I use `setdest` tool to generate **70** nodes in the area `350x350`.

2.1.1 Delay Compare

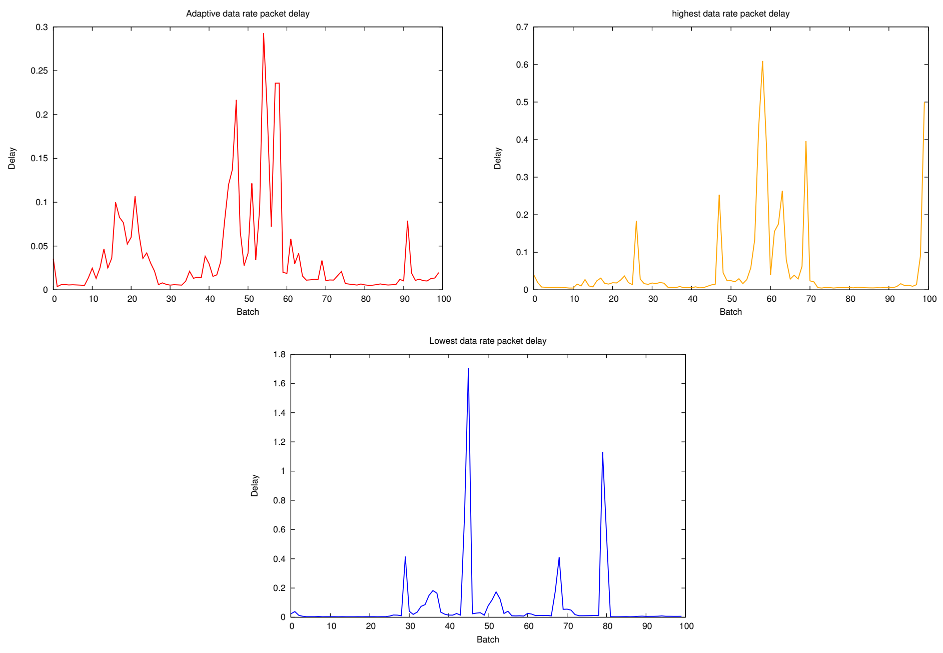

Figure 2.1: Delay Compare

	Avg	Stdev
AD	0.056538	0.144655
HI	0.062663	0.328750
LO	0.155872	1.310392

Table 2.1: Delay Compare

Compare with static model. It is great changed. The average of delay of three case is smaller than previous static model. And the highest end-to-end delay when the delivery rate is lowest. The `AD` case also between `HI` and `LO`. The delay vary sharply range [40,80]. To get the reason for this. I run the `nam`. I found that `node_1` and `node_4` can communicate frequently during the period of time. After this stage, node_1 is far away from node_4 so that they can not communicate.

2.1.2 Delay Jitter Compare

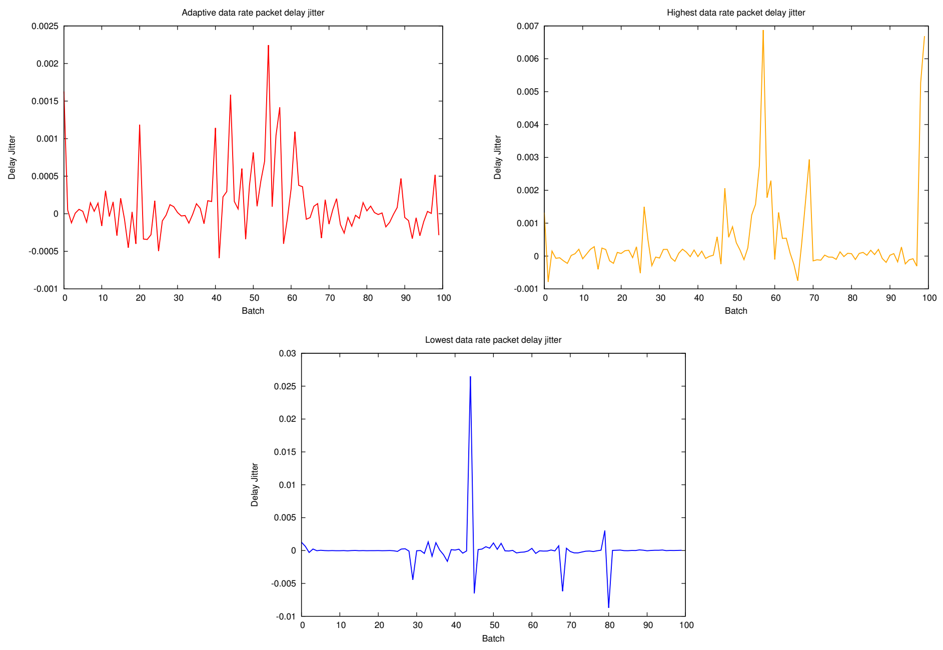

Figure 2.2: Delay jitter Compare

The jitter figure illustrate neatly that during approximately [40,80], node_1 and node_4 peak communicate probability occurs. During the initial stage, node_1 can not get touch with node_4.

2.1.3 Throughput Compare

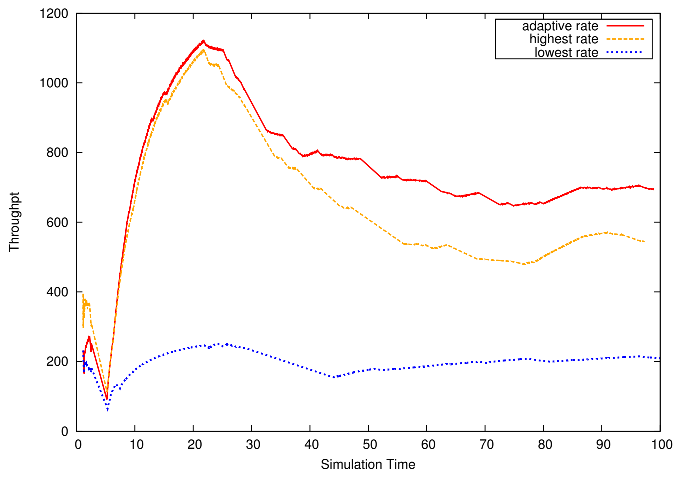

Figure 2.3: Throughput Compare

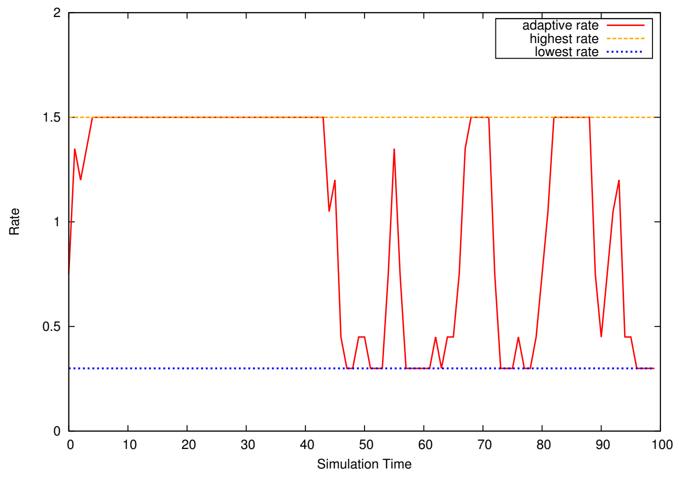

Figure 2.4: Adaptive Rate During Simulation Time

	Avg	Stdev
AD	565.797146	178.759237
HI	749.717684	244.254386
LO	217.068322	25.537143

Table 2.2: Throughput Compare

Obviously, the three case have a general trend. The rank among three case is same to static model. It also is $HI > AD > LO$. Why their vary trend are same with each other. In this scenario, three case share with the same node movement configuration file. So the trend is no different. The other factor also keep same with static model. So the rank is not change.

2.1.4 Loss Rate Compare

	Send Packets	Receive Packets	Loss Rate
AD	7505	5926	0.210393
HI	12154	6320	0.480007
LO	3456	2731	0.209780

Table 2.3: Loss Compare

The highest loss ratio when the delivery rate is `HI`. It is up to 48%. What factor lead to so high loss ratio. What is different by contrast to static scenario? To get the approximate answer, I run `nam`. I found the main difference between these two scenario is that `node_1` may via more than 3 hoc to get communicate with `node_4` in movement scenario. The more hoc, the more probability loss packet.

2.1.5 PSNR Compare

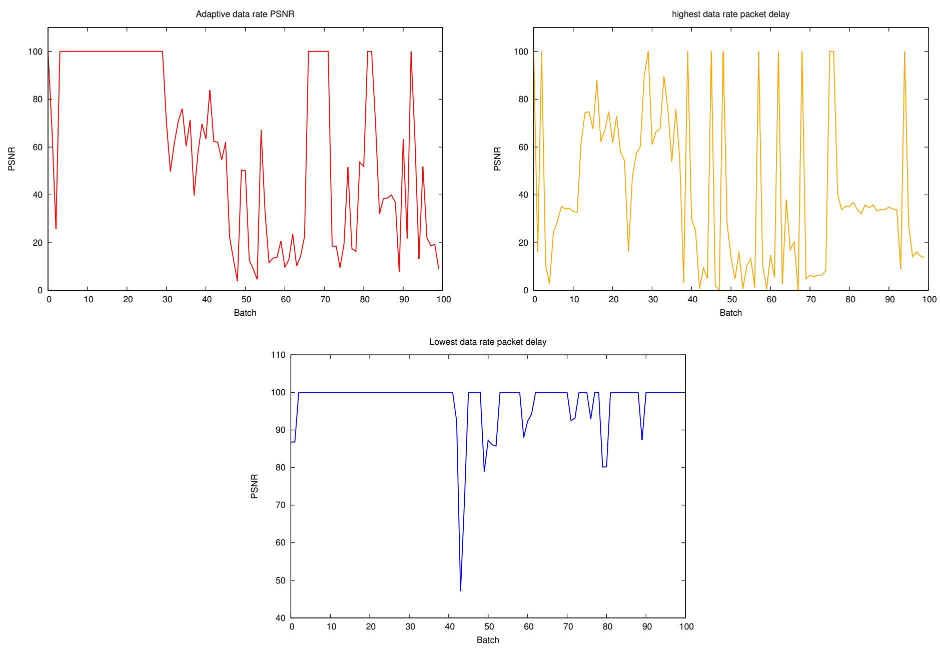

Figure 2.5: PSNR Compare

	Avg	Stdev
AD	94.553861	20.422601
HI	43.137416	35.079797
LO	97.721715	14.298393

Table 2.4: PSNR Compare

Similar to the above performance almost different with static scenario. PSNR also different with static scenario since node is movable. As mention above, node_1 not always can communicate with node_4. Once if the make sense of each other, this stage will keep a short moment. During the stage, the highest transmit maximum data into network. It jitter follow with the relative location of node_1 and node_4.

2.1.6 Summary

Performance / Case	$\overline{Delay}$	$\overline{Throughput}$	$\overline{Loss}$	$\overline{PSNR}$
AD	①	②	②	②
HI	②	①	③	③
LO	③	③	①	①

Table 2.5: Best Performance Sorting

As like static scenario, I give a rank table to compare these three case. The conclusion is keep same, Comprehensive consideration of various factors, I think the `QOAS_AD` is the best solution among three case.

3 Appendix

3.1 TCL simulation script files

Listing 1: 802.11b QOAS Main TCL script

```
proc getopt {argc argv} {

        global opt

        lappend optlist nn

        for {set i 0} {$i < $argc} {incr i} {

            set opt($i) [lindex $argv $i];

        }

}

getopt $argc $argv
set val(scenario) $opt(0)
set val(numnodes) $opt(1)
set val(type) $opt(2)
set val(namfile) $opt(3)
set val(tracefile) $opt(4)

if { $val(scenario) == 2 } {
    set val(movefile) $opt(5)
}

set tip ""
append tip "Load config file:"
append tip $val(type)
puts $tip

proc finish {} {
    global ns_ nf tracefd
    $ns_ flush-trace
    close $nf
    close $tracefd

    exit 0
}

set val(chan) [new Channel/WirelessChannel]
set val(prop) Propagation/TwoRayGround
set val(netif) Phy/WirelessPhy ;
set val(mac) Mac/802_11
set val(ifq) Queue/DropTail/PriQueue ;# interface queue type
set val(ll) LL
set val(ant) Antenna/OmniAntenna ;# antenna model
set val(ifqlen) 50
set val(nn) $val(numnodes)
set val(rp) AODV

set ns_ [new Simulator]
#puts "random [ns-random 0]"

set tracefd [open $val(tracefile) w]
$ns_ trace-all $tracefd
```

```
set nf [open $val(namfile) w]
$ns_ namtrace-all-wireless $nf 350 350

Agent/AODV set ignorehdr_ 1

set topo [new Topography]
$topo load_flatgrid 350 350
set god_ [create-god $val(nn)]

# Pt = 7.21505664 * (d^4) * e-11
# ./threshold -Pt 0.000935073 -fr 2.472e9 -m TwoRayGround 60
Phy/WirelessPhy set CPThresh_ 10.0
Phy/WirelessPhy set CSThresh_ 5.00522e-12
Phy/WirelessPhy set RXThresh_ 2.42253e-11
Phy/WirelessPhy set Pt_ 0.000935073
Phy/WirelessPhy set freq_ 2.472e9
Phy/WirelessPhy set L_ 1.0

Mac/802_11 set CWMin_ 31
Mac/802_11 set CWMax_ 1023
Mac/802_11 set SlotTime_ 0.000020
Mac/802_11 set SIFS_ 0.000010
Mac/802_11 set PreambleLength_ 144
Mac/802_11 set PLCPHeaderLength_ 48
Mac/802_11 set PLCPDataRate_ 1.0e6
Mac/802_11 set dataRate_ 11.0e6
Mac/802_11 set basicRate_ 1.0e6

Mac/802_11 set RTSThreshold_ 256
Mac/802_11 set ShortRetryLimit_ 7
Mac/802_11 set LongRetryLimit_ 4

$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) \
                -channel $val(chan) \
                -topoInstance $topo \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace ON \
                -movementTrace OFF

if { $val(scenario) == 1} {
    for {set i 0} {$i < $val(nn)} {incr i} {
        set node_($i) [$ns_ node]
        $node_($i) random-motion 0
        $node_($i) set X_ [expr 25 + 50*$i]
        $node_($i) set Y_ 175
        $node_($i) set Z_ 0
        $ns_ initial_node_pos $node_($i) 15
    }
} else {
    for {set i 0} {$i < $val(nn)} {incr i} {
        set node_($i) [$ns_ node]
        $node_($i) random-motion 0
    }
    source $val(movefile)

    for {set i 0} {$i < $val(nn)} {incr i} {
        $ns_ initial_node_pos $node_($i) 15
    }

}


# setup a MM UDP connection
set udp_s [new Agent/UDP/UDPmm]
set udp_r [new Agent/UDP/UDPmm]
$ns_ attach-agent $node_(0) $udp_s
$ns_ attach-agent $node_(3) $udp_r
$ns_ connect $udp_s $udp_r
$udp_s set packetSize_ 1000
$udp_r set packetSize_ 1000
$udp_s set fid_ 1
$udp_r set fid_ 1

#setup a MM Application
set qoas_s [new Application/QOASServer]
set qoas_r [new Application/QOASClient]

source $val(type)

$qoas_s attach-agent $udp_s
$qoas_r attach-agent $udp_r
$qoas_s set pktsize_ 1000
$qoas_s set random_ false


set udpcbrs [new Agent/UDP]
$udpcbrs set fid_ 2
$udpcbrs set packetsize_ 1000
$ns_ attach-agent $node_(0) $udpcbrs

set udpcbrr [new Agent/UDP]
$ns_ attach-agent $node_(3) $udpcbrr
$ns_ connect $udpcbrs $udpcbrr


set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udpcbrs
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 0.5mb
$cbr set random_ false


$ns_ at 1.0 "$cbr start"
$ns_ at 1.0 "$qoas_s start"
$ns_ at 100 "finish"
puts "Starting Simulation..."

$ns_ run
```

3.2 AWK analysis script file

Listing 2: Analysis throughput,delay,delay jitter,loss rate

```
BEGIN {
    highest_pkt_id = 0;
    total_delay = 0;
    send_pkt = 0;
    recv_pkt = 0;
    drop_pkt = 0;
    i = 0;
    begin_time = -1;
}

$0 {
    layer = $4;
    pkt_type = $7;

    if(layer == "MAC" && pkt_type == application) {
        action = $1;
        time = $2;
        pkt_id = $6;
        pkt_size = $8;

        crt_node_id = $3;

        if( action=="s" && crt_node_id == "_0_" )
        {
            send_pkt += 1;

            if( pkt_id > highest_pkt_id)
                highest_pkt_id = pkt_id

            if( start_time[pkt_id] == 0 )
                start_time[pkt_id] = time;
        }


        if( action=="r" && crt_node_id == "_3_" )
        {
            recv_pkt += 1;

            pkt_size_sum[i+1] = pkt_size_sum[i] + pkt_size;

            if(begin_time == -1)  begin_time = time

            end_time[pkt_id] = time;

            over_time[i] = time;
            i = i+1;
        }
    }
}


END {

    last_seqno = 0;
    last_delay = 0;
    seqno_diff = 0;

    for (pkt_id = 0; pkt_id <= highest_pkt_id;pkt_id++)
```

```
        {
            start = start_time[pkt_id];
            end = end_time[pkt_id];
            duration = end - start;

            if(start < end)
            {
                total_delay += duration;
                printf("%d\t%f\n", pkt_id, duration) >> delayfile;


                seqno_diff = pkt_id - last_seqno;
                delay_diff = duration - last_delay;

                if ( seqno_diff == 0)
                    jitter = 0;
                else
                    jitter = delay_diff / seqno_diff;

                printf("%d\t%f\n",pkt_id,jitter) >> jitterfile;
                last_seqno=pkt_id;
                last_delay=duration;
            }
        }

        for(j=2;j<i;j++)
        {
            throughput = pkt_size_sum[j]/(over_time[j]-begin_time)
                *8/1000;
            printf("%.2f\t%.2f\n",over_time[j],throughput) >>
                throughputfile;
        }

        printf("send %d packets,receive %d packets,loss %d packets,loss
            rate: %f\n",
                send_pkt,recv_pkt,
                send_pkt-recv_pkt,
                (send_pkt-recv_pkt)/send_pkt) >> summaryfile;


        printf("recv_pkt: %d, average delay: %f\n",
               recv_pkt,
               total_delay/recv_pkt) >> summaryfile;

    }
```

3.3 Build script file

Listing 3: Simulation,Draw datagram and Build report document

```
# clear workspace
if [ -d "trace" ]; then
    rm -rf trace/
fi

if [ -d "result" ]; then
    rm -rf result/
fi

if [ -d "log" ]; then
    rm -rf log/
fi

# rebuild workspace

mkdir -p trace/scenario_1/udp
mkdir -p trace/scenario_1/cbr
mkdir -p trace/scenario_2/udp
mkdir -p trace/scenario_2/cbr

mkdir -p result/scenario_1/udp
mkdir -p result/scenario_1/cbr
mkdir -p result/scenario_2/udp
mkdir -p result/scenario_2/cbr

mkdir -p log/scenario_1/udp
mkdir -p log/scenario_1/cbr
mkdir -p log/scenario_2/udp
mkdir -p log/scenario_2/cbr

# start to simulate scenario I
echo "start to simulate scenario I \n"
ns main.tcl 1 4 ad.tcl ad.nam ad.tr >> log/scenario_1/ad.log
ns main.tcl 1 4 hi.tcl hi.nam hi.tr >> log/scenario_1/hi.log
ns main.tcl 1 4 lo.tcl lo.nam lo.tr >> log/scenario_1/lo.log
echo "finish simulating scenario I \n"

# start to analysis scenario I trace file
echo "start to analysis scenario I trace file\n"

sh analysis.sh udp ad 1
sh analysis.sh udp hi 1
sh analysis.sh udp lo 1

sh analysis.sh cbr ad 1
sh analysis.sh cbr hi 1
sh analysis.sh cbr lo 1

echo "finish analysing scenario I trace file\n"

# move nam and tr file to corresponding directory
mv *.nam trace/scenario_1
mv *.tr trace/scenario_1

node_num=70
#sh setdest.sh $node_num

# start to simulate scenario II
```

```
echo "start to simulate scenario II\n"
ns main.tcl 2 $node_num ad.tcl ad.nam ad.tr move.tcl >> log/
    scenario_2/ad.log
ns main.tcl 2 $node_num hi.tcl hi.nam hi.tr move.tcl >> log/
    scenario_2/hi.log
ns main.tcl 2 $node_num lo.tcl lo.nam lo.tr move.tcl >> log/
    scenario_2/lo.log
echo "finish simulating scenario II\n"

echo "start to analysis scenario II trace file\n"

sh analysis.sh udp ad 2
sh analysis.sh udp hi 2
sh analysis.sh udp lo 2

sh analysis.sh cbr ad 2
sh analysis.sh cbr hi 2
sh analysis.sh cbr lo 2

echo "finishing analysing scenario II trace file\n"

# move nam and tr file to corresponding directory
mv *.nam trace/scenario_2
mv *.tr trace/scenario_2

echo "finish all job\n"
```

3.4 Readme

how to run this simulation
Assume you have install all follow software:

1. `ns-2.34`
2. `gnuplot`
3. `texlive 2010 suit`

Note: you should patch the `qoas.patch` and `qoas_chang.patch` to `ns-2.34`. These two patch reside in `patch` directory.

step to build
firstly, change work directory to current directory

1. run `sh run.sh` to get the simulation result.
2. cd to `doc` directory, type the command: `cd ./doc`
3. and then run `sh doc.sh` to build the report document base on the result.

If you don't want to re-build this report document. Ignore step 2 and 3. To view the output result. You can open the result directory. All the result gathering here. After run the main script `run.sh`. You can change directory to `trace` to view the `trace` file and `nam` file.

All analysis `awk` files put into the `awk` directory.