How to measure packet loss rate, jitter, and end-to-end delay for UDP-based applications?

[**TCL script**]

```
#Create a simulator object
set ns [new Simulator]
#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red
#Open the NAM trace file
set nf [open out.nam w]
$ns namtrace-all $nf
#Open the traffic trace file to record all events
set nd [open out.tr w]
$ns trace-all $nd
#Define a ‘finish’ procedure
proc finish {} {
global ns nf nd
$ns flush-trace
#Close the NAM trace file
close $nf
close $nd
#Execute NAM on the trace file
exec nam out.nam &
exit 0
}
#Create four nodes
set n0 [$ns node]
```

```
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
#Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail
#Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 10
#Give node position (for NAM)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right
#Monitor the queue for link (n2-n3). (for NAM)
$ns duplex-link-op $n2 $n3 queuePos 0.5
#Setup a TCP connection
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
#Setup a UDP connection
```

```
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2
#Setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
$cbr set random_ false
#Schedule events for the CBR and FTP agents
$ns at 0.1 “$cbr start”
$ns at 1.0 “$ftp start”
$ns at 4.0 “$ftp stop”
$ns at 4.5 “$cbr stop”
#Detach tcp and sink agents (not really necessary)
$ns at 4.5 “$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink”
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 “finish”
#Print CBR packet size and interval
puts “CBR packet size = [$cbr set packet_size_]”
puts “CBR interval = [$cbr set interval_]”
#Run the simulation
$ns run
```

[**First method—Parsing the traffic trace file**]

After simulation, you will get a traffic trace file “out.tr”. Then I will show how to use gawk to

parse the out.tr to get packet loss rate, jitter, and end-to-end delay.

[**Packet Loss Rate**]

(FileName: measure-loss.awk)

```
#This program is used to calculate the packet loss rate for CBR program
BEGIN {
# Initialization. Set two variables. fsDrops: packets drop. numFs: packets sent
fsDrops = 0;
numFs = 0;
}
{
action = $1;
time = $2;
from = $3;
to = $4;
type = $5;
pktsize = $6;
flow_id = $8;
src = $9;
dst = $10;
seq_no = $11;
packet_id = $12;
if (from==1 && to==2 && action == “+”)
numFs++;
if (flow_id==2 && action == “d”)
fsDrops++;
}
END {
printf(“number of packets sent:%d lost:%d\n”, numFs, fsDrops);
```

```
}
```

Usage:

$gawk –f measure-loss.awk out.tr

Results:

number of packets sent:550 lost:8

Description:

From the results, we know that sender sent 550 packets, but 8 packets got lost.

[**Jitter**]

(FileName: measure-jitter.awk)

```
#This program is used to calculate the jitters for CBR
# jitter  = ((recvtime(j)-sendtime(j))-(recvtime(i)-sendtime(i)))/(j-i), j > i
BEGIN {
# Initialization
highest_packet_id = 0;
}
{
action = $1;
time = $2;
from = $3;
to = $4;
type = $5;
pktsize = $6;
flow_id = $8;
src = $9;
dst = $10;
seq_no = $11;
packet_id = $12;
if ( packet_id > highest_packet_id ) {
```

```
highest_packet_id = packet_id;
}
#Record the transmission time
if ( start_time[packet_id] == 0 ) {
# Record the sequence number
pkt_seqno[packet_id] = seq_no;
start_time[packet_id] = time;
}
#Record the receiving time for CBR (flow_id=2)
if ( flow_id == 2 && action != “d” ) {
if ( action == “r” ) {
end_time[packet_id] = time;
}
} else {
end_time[packet_id] = -1;
}
}
END {
last_seqno = 0;
last_delay = 0;
seqno_diff = 0;
for ( packet_id = 0; packet_id <= highest_packet_id; packet_id++ ) {
start = start_time[packet_id];
end = end_time[packet_id];
packet_duration = end – start;
if ( start < end ) {
seqno_diff = pkt_seqno[packet_id] – last_seqno;
delay_diff = packet_duration – last_delay;
```

```
if (seqno_diff == 0) {
jitter =0;
} else {
jitter = delay_diff/seqno_diff;
}
printf("%f %f\n", start, jitter);
last_seqno = pkt_seqno[packet_id];
last_delay = packet_duration;
}
}
}
```

Usage:

$gawk –f measure-jitter.awk out.tr

Results:

0.100000 0.000000

0.108000 0.000000

0.116000 0.000000

……

Description:

You will the jitter of initial records is 0. This is because there is only CBR traffic flow, no other background traffic flow. After FTP flow starts, you can see as follows.

1.068000 0.002296

1.076000 0.001600

1.084000 -0.003294

1.092000 -0.000602

……

[**End-to-End Delay**]

(FileName: measure-delay.awk)

```
#This program is used to calculate the end-to-end delay for CBR
```

```
BEGIN {
highest_packet_id = 0;
}
{
action = $1;
time = $2;
from = $3;
to = $4;
type = $5;
pktsize = $6;
flow_id = $8;
src = $9;
dst = $10;
seq_no = $11;
packet_id = $12;
if ( packet_id > highest_packet_id )
highest_packet_id = packet_id;
if ( start_time[packet_id] == 0 )
start_time[packet_id] = time;
if ( flow_id == 2 && action != “d” ) {
if ( action == “r” ) {
end_time[packet_id] = time;
}
} else {
end_time[packet_id] = -1;
}
}
END {
```

```
for ( packet_id = 0; packet_id <= highest_packet_id; packet_id++ ) {
start = start_time[packet_id];
end = end_time[packet_id];
packet_duration = end – start;
if ( start < end ) printf("%f %f\n", start, packet_duration);
}
}
```

Usage:

$gawk –f measure-delay.awk out.tr

Results:

0.100000 0.038706

0.108000 0.038706

0.116000 0.038706

……

[**Method 2—modify C++ code**]

The basic idea is to insert two fields, sendtime_ and pkt_id_, in the hdr_cmn header. When packets are sent, the packet id and send time is recorded in the sender trace file. Then when packets are received at the destination, the packet id and receiving time is recorded in the receiver trace file. So I prepare two agents, mudp and mudpsink to do the jobs. Mudp is the extension of udp agent. It only overrides the sendmsg function to keep download the packet id and sendtime in the user specified file.

[**Insert the codes into NS2**]

1. Download **mudp.cc**, **mudp.h**, **mudpsink.cc**, and **mudpsink.h**.

2. Create a folder named measure under ns. (for example, ~/ns-allinone-2.28/ns-2.28/measure)

3. Put these four files into measure folder.

4. add sendtime_, pkt_id_ into packet common header. (modify common/packet.h)

```
struct hdr_cmn {
enum dir_t { DOWN= -1, NONE= 0, UP= 1 };
packet_t ptype_; // packet type (see above)
int size_; // simulated packet size
int uid_; // unique id
```

int error_; // error flag

int errbitcnt_; // # of corrupted bits jahn

int fecsize_;

double ts_; // timestamp: for q-delay measurement

int iface_; // receiving interface (label)

dir_t direction_; // direction: 0=none, 1=up, -1=down

double sendtime_; …

unsigned long int pkt_id_;

…

inline int& addr_type() { return (addr_type_); }

inline int& num_forwards() { return (num_forwards_); }

inline int& opt_num_forwards() { return (opt_num_forwards_); }

//monarch_end

inline double& sendtime() { return (sendtime_); } // added by smallko

}

5. Add the “measure/mudp.o measure/mudpsink.o “ in the OBJ_CC of Makefile.

6. Add “Agent/mUDP set packetSize_ 1000” in the ns-default.tcl. (ns-default.tcl is under ~/ns-allinone-2.28/ns-2.28/tcl/lib)

7. make clean ; make

[**TCL script**]

```
#Create a simulator object

set ns [new Simulator]

#Define different colors for data flows (for NAM)

$ns color 1 Blue

$ns color 2 Red

#Open the NAM trace file

set nf [open out.nam w]

$ns namtrace-all $nf

#Open the traffic trace file to record all events
```

```
set nd [open out.tr w]
$ns trace-all $nd
#Define a ‘finish’ procedure
proc finish {} {
global ns nf nd
$ns flush-trace
#Close the NAM trace file
close $nf
close $nd
#Execute NAM on the trace file
#exec nam out.nam &
exit 0
}
#Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
#Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail
#Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 10
#Give node position (for NAM)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right
```

```
#Monitor the queue for link (n2-n3). (for NAM)
$ns duplex-link-op $n2 $n3 queuePos 0.5
#Setup a TCP connection
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
#Setup a mUDP connection
set udp [new Agent/mUDP]
#set the sender trace file name (sd)
$udp set_filename sd
$ns attach-agent $n1 $udp
set null [new Agent/mUdpSink]
#set the receiver trace file name (rd)
$null set_filename rd
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2
#Setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
```

```
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
$cbr set random_ false
#Schedule events for the CBR and FTP agents
$ns at 0.1 “$cbr start”
$ns at 1.0 “$ftp start”
$ns at 4.0 “$ftp stop”
$ns at 4.5 “$cbr stop”
#Detach tcp and sink agents (not really necessary)
$ns at 4.5 “$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink”
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 “finish”
#Print CBR packet size and interval
puts “CBR packet size = [$cbr set packet_size_]”
puts “CBR interval = [$cbr set interval_]”
#Run the simulation
$ns run
```

After running, you will get sd and rd.

(sd)

```
0 0.100000
1 0.108000
2 0.116000
3 0.124000
4 0.132000
…….
```

The first column: packet id ; the second column: packet send time

(rd)

0 0.100000 0.138706 0.038706

1 0.108000 0.146706 0.038706

2 0.116000 0.154706 0.038706

……

The first column: packet id ; the second column: packet send time; the third column: packet receiving time; the fourth column: end-to-end delay.