LAMPIRAN

LAMPIRAN A

SKRIP SIMULASI TFMCC

SKRIP SIMULASI TFMCC

1. VERSI JORG WIDMER

```
set interval 1.0
set duration 120.0
set psize 1000
set fid 0

set tcp_num 0
set tfmcc_num 0
set tfmcc_recv_num 0

ns-random 0

# better if bw and packet size don't match exactly (i.e., exactly one
# packet each 10ms), otherwise you might sometimes see strange
# synchronization effects in ns!
set bw 1007
set num_links 4

Agent/TCP set packetSize_ $psize
Agent/TFMCC set packetSize_ $psize

# always add some randomness, s.o.
Agent/TCP set overhead_ 0.002

proc setupTCP {start stop s r} {
   global ns tcp_num fid interval tcp tcp_sink ftp
   set tcp($tcp_num) [new Agent/TCP/Sack1]
   $tcp($tcp_num) set fid_ $fid
   set tcp_sink($tcp_num) [new Agent/TCPSink/Sack1]
   $ns attach-agent $s $tcp($tcp_num)
   $ns attach-agent $r $tcp_sink($tcp_num)
   $ns connect $tcp($tcp_num) $tcp_sink($tcp_num)
   set ftp($tcp_num) [new Application/FTP]
   $ftp($tcp_num) attach-agent $tcp($tcp_num)
   $ns at $start "$ftp($tcp_num) start"
   $ns at $stop "$ftp($tcp_num) stop"
   incr tcp_num
   incr fid
   return $tcp([expr $tcp_num-1])
}
```

```
proc setupTFMCC {start stop s} {
   global ns tfmcc_num fid interval tfmcc
   set group [Node allocaddr]
   set tfmcc($tfmcc_num) [new Agent/TFMCC]
   $tfmcc($tfmcc_num) set fid_ $fid
   $tfmcc($tfmcc_num) set dst_addr_ $group
   $tfmcc($tfmcc_num) set dst_port_ 0
   $ns attach-agent $s $tfmcc($tfmcc_num)
   $ns at $start "$tfmcc($tfmcc_num) start"
   $ns at $stop "$tfmcc($tfmcc_num) stop"
   incr tfmcc_num
   incr fid
   return $tfmcc([expr $tfmcc_num-1])
}

proc setupTFMCCSink {start stop t r} {
   global ns tfmcc_recv_num tfmcc_sink
   set group [$t set dst_addr_]
   set tfmcc_sink($tfmcc_recv_num) [new Agent/TFMCCSink]
   $ns attach-agent $r $tfmcc_sink($tfmcc_recv_num)
   $tfmcc_sink($tfmcc_recv_num) set dst_addr_ [$t set agent_addr_]
   $tfmcc_sink($tfmcc_recv_num) set dst_port_ [$t set agent_port_]
   $tfmcc_sink($tfmcc_recv_num) set recv_id_ $tfmcc_recv_num
   $ns at $start "$r join-group $tfmcc_sink($tfmcc_recv_num) $group"
   $ns at $stop "$tfmcc_sink($tfmcc_recv_num) stop; $r leave-group
$tfmcc_sink($tfmcc_recv_num) $group"
   incr tfmcc_recv_num
   return $tfmcc_sink([expr $tfmcc_recv_num-1])
}

set ns [new Simulator -multicast on]

#set f [open output/$bw.pkt w]
#$ns trace-all $f

set n(0) [$ns node]
set n(1) [$ns node]
for {set i 0} {$i < $num_links} {incr i} {
   set s($i) [$ns node]
   set r($i) [$ns node]
}
#$ns duplex-link $n(0) $n(1) [expr $bw*1000] 20ms RED
$ns duplex-link $n(0) $n(1) [expr $bw*1000] 20ms DropTail
for {set i 0} {$i < $num_links} {incr i} {
```

```
    $ns duplex-link $s($i) $n(0) 100Mb 5ms DropTail
    $ns duplex-link $n(1) $r($i) 100Mb 5ms DropTail
}

set mproto DM
set mrthandle [$ns mrtproto $mproto {}]

set t [setupTFMCC 1.0 $duration $s(0)]
for {set i 0} {$i < $num_links} {incr i} {
    setupTFMCCSink 0.9 $duration $t $r($i)
}
for {set i 0} {$i < $num_links} {incr i} {
    setupTCP 2.0 $duration $s($i) $r($i)
}

$ns at $duration "exit 0"
$ns run
```

2. UNTUK VARIABEL BANDWIDTH

```
set interval 1.0
set duration 120.0
set psize 1000
set fid 0

set tcp_num 0
set tfmcc_num 0
set tfmcc_recv_num 0

ns-random 0

# better if bw and packet size don't match exactly (i.e., exactly one
# packet each 10ms), otherwise you might sometimes see strange
# synchronization effects in ns!
set bw 250
# dapat diubah menjadi BW 500, 1000, 1500, dan 2000 kilobytes per second.
set num_links 4

Agent/TCP set packetSize_ $psize
Agent/TFMCC set packetSize_ $psize

# always add some randomness, s.o.
Agent/TCP set overhead_ 0.002
```

```
proc setupTCP {start stop s r} {
    global ns tcp_num fid interval tcp tcp_sink ftp
    set tcp($tcp_num) [new Agent/TCP/Sack1]
    $tcp($tcp_num) set fid_ $fid
    set tcp_sink($tcp_num) [new Agent/TCPSink/Sack1]
    $ns attach-agent $s $tcp($tcp_num)
    $ns attach-agent $r $tcp_sink($tcp_num)
    $ns connect $tcp($tcp_num) $tcp_sink($tcp_num)
    set ftp($tcp_num) [new Application/FTP]
    $ftp($tcp_num) attach-agent $tcp($tcp_num)
    $ns at $start "$ftp($tcp_num) start"
    $ns at $stop "$ftp($tcp_num) stop"
    incr tcp_num
    incr fid
    return $tcp([expr $tcp_num-1])
}

proc setupTFMCC {start stop s} {
    global ns tfmcc_num fid interval tfmcc
    set group [Node allocaddr]
    set tfmcc($tfmcc_num) [new Agent/TFMCC]
    $tfmcc($tfmcc_num) set fid_ $fid
    $tfmcc($tfmcc_num) set dst_addr_ $group
    $tfmcc($tfmcc_num) set dst_port_ 0
    $ns attach-agent $s $tfmcc($tfmcc_num)
    $ns at $start "$tfmcc($tfmcc_num) start"
    $ns at $stop "$tfmcc($tfmcc_num) stop"
    incr tfmcc_num
    incr fid
    return $tfmcc([expr $tfmcc_num-1])
}

proc setupTFMCCSink {start stop t r} {
    global ns tfmcc_recv_num tfmcc_sink
    set group [$t set dst_addr_]
    set tfmcc_sink($tfmcc_recv_num) [new Agent/TFMCCSink]
    $ns attach-agent $r $tfmcc_sink($tfmcc_recv_num)
    $tfmcc_sink($tfmcc_recv_num) set dst_addr_ [$t set agent_addr_]
    $tfmcc_sink($tfmcc_recv_num) set dst_port_ [$t set agent_port_]
    $tfmcc_sink($tfmcc_recv_num) set recv_id_ $tfmcc_recv_num
    $ns at $start "$r join-group $tfmcc_sink($tfmcc_recv_num) $group"
    $ns at $stop "$tfmcc_sink($tfmcc_recv_num) stop; $r leave-group
$tfmcc_sink($tfmcc_recv_num) $group"
    incr tfmcc_recv_num
```

```
    return $tfmcc_sink([expr $tfmcc_recv_num-1])
}

global ns
set ns [new Simulator -multicast on]

set f0 [open out_bottleneck.tr w]
$ns trace-all $f0
set f1 [open out_bottleneck.nam w]
$ns namtrace-all $f1
puts " Simulation started."

# Declare "finish" procedure which closes the trace file
proc finish {} {
        global ns f0 f1
        $ns flush-trace
        close $f0
        close $f1
        exec nam out_bottleneck.nam &
        puts " Simulation ended."
        exit 0
}

set n(0) [$ns node]
set n(1) [$ns node]
for {set i 0} {$i < $num_links} {incr i} {
    set s($i) [$ns node]
    set r($i) [$ns node]
}
#$ns duplex-link $n(0) $n(1) [expr $bw*1000] 20ms RED
$ns duplex-link $n(0) $n(1) [expr $bw*1000] 20ms DropTail
for {set i 0} {$i < $num_links} {incr i} {
    $ns duplex-link $s($i) $n(0) 100Mb 5ms DropTail
    $ns duplex-link $n(1) $r($i) 100Mb 5ms DropTail
}

set mproto DM
set mrthandle [$ns mrtproto $mproto {}]

set t [setupTFMCC 1.0 $duration $s(0)]
for {set i 0} {$i < $num_links} {incr i} {
    setupTFMCCSink 0.9 $duration $t $r($i)
}
for {set i 0} {$i < $num_links} {incr i} {
    setupTCP 2.0 $duration $s($i) $r($i)
```

```
}

$ns at $duration "finish"
$ns run
```

3. UNTUK VARIABEL PACKET SIZE

```
set interval 1.0
set duration 120.0
set psize 250
# dapat diubah menjadi packet size 500, 1000, 1500, dan 2000 bytes.
set fid 0

set tcp_num 0
set tfmcc_num 0
set tfmcc_recv_num 0

ns-random 0

# better if bw and packet size don't match exactly (i.e., exactly one
# packet each 10ms), otherwise you might sometimes see strange
# synchronization effects in ns!
set bw 1007
set num_links 4

Agent/TCP set packetSize_ $psize
Agent/TFMCC set packetSize_ $psize

# always add some randomness, s.o.
Agent/TCP set overhead_ 0.002

proc setupTCP {start stop s r} {
   global ns tcp_num fid interval tcp tcp_sink ftp
   set tcp($tcp_num) [new Agent/TCP/Sack1]
   $tcp($tcp_num) set fid_ $fid
   set tcp_sink($tcp_num) [new Agent/TCPSink/Sack1]
   $ns attach-agent $s $tcp($tcp_num)
   $ns attach-agent $r $tcp_sink($tcp_num)
   $ns connect $tcp($tcp_num) $tcp_sink($tcp_num)
   set ftp($tcp_num) [new Application/FTP]
   $ftp($tcp_num) attach-agent $tcp($tcp_num)
   $ns at $start "$ftp($tcp_num) start"
   $ns at $stop "$ftp($tcp_num) stop"
```

```
    incr tcp_num
    incr fid
    return $tcp([expr $tcp_num-1])
}

proc setupTFMCC {start stop s} {
    global ns tfmcc_num fid interval tfmcc
    set group [Node allocaddr]
    set tfmcc($tfmcc_num) [new Agent/TFMCC]
    $tfmcc($tfmcc_num) set fid_ $fid
    $tfmcc($tfmcc_num) set dst_addr_ $group
    $tfmcc($tfmcc_num) set dst_port_ 0
    $ns attach-agent $s $tfmcc($tfmcc_num)
    $ns at $start "$tfmcc($tfmcc_num) start"
    $ns at $stop "$tfmcc($tfmcc_num) stop"
    incr tfmcc_num
    incr fid
    return $tfmcc([expr $tfmcc_num-1])
}

proc setupTFMCCSink {start stop t r} {
    global ns tfmcc_recv_num tfmcc_sink
    set group [$t set dst_addr_]
    set tfmcc_sink($tfmcc_recv_num) [new Agent/TFMCCSink]
    $ns attach-agent $r $tfmcc_sink($tfmcc_recv_num)
    $tfmcc_sink($tfmcc_recv_num) set dst_addr_ [$t set agent_addr_]
    $tfmcc_sink($tfmcc_recv_num) set dst_port_ [$t set agent_port_]
    $tfmcc_sink($tfmcc_recv_num) set recv_id_ $tfmcc_recv_num
    $ns at $start "$r join-group $tfmcc_sink($tfmcc_recv_num) $group"
    $ns at $stop "$tfmcc_sink($tfmcc_recv_num) stop; $r leave-group
$tfmcc_sink($tfmcc_recv_num) $group"
    incr tfmcc_recv_num
    return $tfmcc_sink([expr $tfmcc_recv_num-1])
}

global ns
set ns [new Simulator -multicast on]

set f0 [open out_bottleneck.tr w]
$ns trace-all $f0
set f1 [open out_bottleneck.nam w]
$ns namtrace-all $f1
puts " Simulation started."

# Declare "finish" procedure which closes the trace file
```

```
proc finish {} {
        global ns f0 f1
        $ns flush-trace
        close $f0
        close $f1
        exec nam out_bottleneck.nam &
        puts " Simulation ended."
        exit 0
}

set n(0) [$ns node]
set n(1) [$ns node]
for {set i 0} {$i < $num_links} {incr i} {
   set s($i) [$ns node]
   set r($i) [$ns node]
}
#$ns duplex-link $n(0) $n(1) [expr $bw*1000] 20ms RED
$ns duplex-link $n(0) $n(1) [expr $bw*1000] 20ms DropTail
for {set i 0} {$i < $num_links} {incr i} {
   $ns duplex-link $s($i) $n(0) 100Mb 5ms DropTail
   $ns duplex-link $n(1) $r($i) 100Mb 5ms DropTail
}

set mproto DM
set mrthandle [$ns mrtproto $mproto {}]

set t [setupTFMCC 1.0 $duration $s(0)]
for {set i 0} {$i < $num_links} {incr i} {
   setupTFMCCSink 0.9 $duration $t $r($i)
}
for {set i 0} {$i < $num_links} {incr i} {
   setupTCP 2.0 $duration $s($i) $r($i)
}

$ns at $duration "finish"
$ns run
```

4. UNTUK VARIABEL JUMLAH PENERIMA

```
set interval 1.0
set duration 120.0
set psize 1000
set fid 0
```

```
set tcp_num 0
set tfmcc_num 0
set tfmcc_recv_num 0

ns-random 0

# better if bw and packet size don't match exactly (i.e., exactly one
# packet each 10ms), otherwise you might sometimes see strange
# synchronization effects in ns!
set bw 1007
```
#num_links dihilangkan, supaya jumlah pengirim dapat berbeda dengan jumlah penerima

```
Agent/TCP set packetSize_ $psize
Agent/TFMCC set packetSize_ $psize

# always add some randomness, s.o.
Agent/TCP set overhead_ 0.002

proc setupTCP {start stop s r} {
    global ns tcp_num fid interval tcp tcp_sink ftp
    set tcp($tcp_num) [new Agent/TCP/Sack1]
    $tcp($tcp_num) set fid_ $fid
    set tcp_sink($tcp_num) [new Agent/TCPSink/Sack1]
    $ns attach-agent $s $tcp($tcp_num)
    $ns attach-agent $r $tcp_sink($tcp_num)
    $ns connect $tcp($tcp_num) $tcp_sink($tcp_num)
    set ftp($tcp_num) [new Application/FTP]
    $ftp($tcp_num) attach-agent $tcp($tcp_num)
    $ns at $start "$ftp($tcp_num) start"
    $ns at $stop "$ftp($tcp_num) stop"
    incr tcp_num
    incr fid
    return $tcp([expr $tcp_num-1])
}

proc setupTFMCC {start stop s} {
    global ns tfmcc_num fid interval tfmcc
    set group [Node allocaddr]
    set tfmcc($tfmcc_num) [new Agent/TFMCC]
    $tfmcc($tfmcc_num) set fid_ $fid
    $tfmcc($tfmcc_num) set dst_addr_ $group
    $tfmcc($tfmcc_num) set dst_port_ 0
    $ns attach-agent $s $tfmcc($tfmcc_num)
```

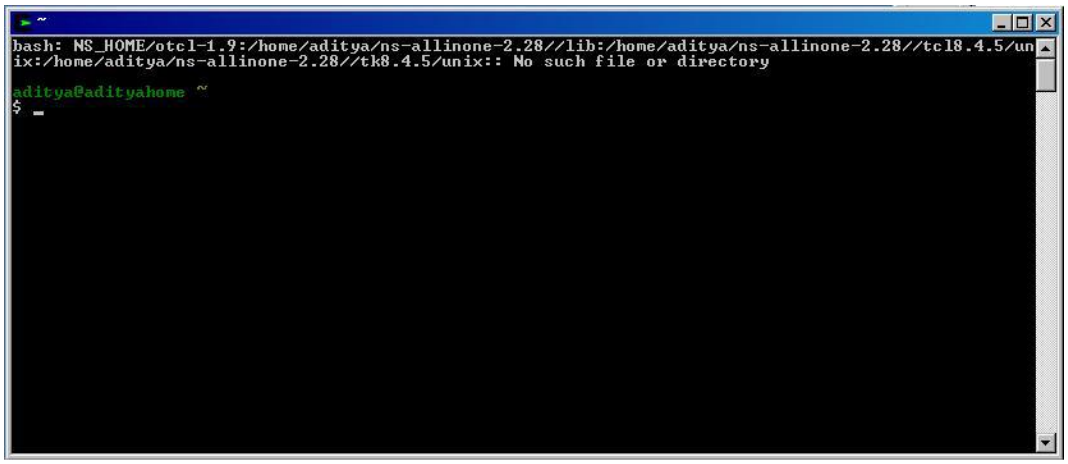
```
    $ns at $start "$tfmcc($tfmcc_num) start"
    $ns at $stop "$tfmcc($tfmcc_num) stop"
    incr tfmcc_num
    incr fid
    return $tfmcc([expr $tfmcc_num-1])
}

proc setupTFMCCSink {start stop t r} {
    global ns tfmcc_recv_num tfmcc_sink
    set group [$t set dst_addr_]
    set tfmcc_sink($tfmcc_recv_num) [new Agent/TFMCCSink]
    $ns attach-agent $r $tfmcc_sink($tfmcc_recv_num)
    $tfmcc_sink($tfmcc_recv_num) set dst_addr_ [$t set agent_addr_]
    $tfmcc_sink($tfmcc_recv_num) set dst_port_ [$t set agent_port_]
    $tfmcc_sink($tfmcc_recv_num) set recv_id_ $tfmcc_recv_num
    $ns at $start "$r join-group $tfmcc_sink($tfmcc_recv_num) $group"
    $ns at $stop "$tfmcc_sink($tfmcc_recv_num) stop; $r leave-group
$tfmcc_sink($tfmcc_recv_num) $group"
    incr tfmcc_recv_num
    return $tfmcc_sink([expr $tfmcc_recv_num-1])
}

global ns
set ns [new Simulator -multicast on]


set f0 [open out_bottleneck.tr w]
$ns trace-all $f0
set f1 [open out_bottleneck.nam w]
$ns namtrace-all $f1
puts " Simulation started."

# Declare "finish" procedure which closes the trace file
proc finish {} {
        global ns f0 f1
        $ns flush-trace
        close $f0
        close $f1
        exec nam out_bottleneck.nam &
        puts " Simulation ended."
        exit 0
}

set n(0) [$ns node]
set n(1) [$ns node]
```

```
for {set i 0} {$i < 2} {incr i} {
   set s($i) [$ns node]
}
#jumlah node pengirim
for {set i 0} {$i < 2} {incr i} {
   set r($i) [$ns node]
}
#jumlah node penerima
$ns duplex-link $n(0) $n(1) 1Mb 20ms DropTail

for {set i 0} {$i < 2} {incr i} {
   $ns duplex-link $s($i) $n(0) 100Mb 5ms DropTail
}
for {set i 0} {$i < 2} {incr i} {
   $ns duplex-link $n(1) $r($i) 100Mb 5ms DropTail
}

# Set Nodes Orientation
$ns duplex-link-op $s(0) $n(0) orient right-down
$ns duplex-link-op $s(1) $n(0) orient right-up

$ns duplex-link-op $n(0) $n(1) orient right

$ns duplex-link-op $n(1) $r(0) orient right-up
$ns duplex-link-op $n(1) $r(1) orient right-down

set mproto DM
set mrthandle [$ns mrtproto $mproto {}]

set t [setupTFMCC 1.0 $duration $s(0)]
for {set i 0} {$i < 2} {incr i} {
   setupTFMCCSink 0.9 $duration $t $r($i)
}
for {set i 0} {$i < 2} {incr i} {
   setupTCP 2.0 $duration $s(1) $r($i)
}

$ns at $duration "finish"
$ns run
```

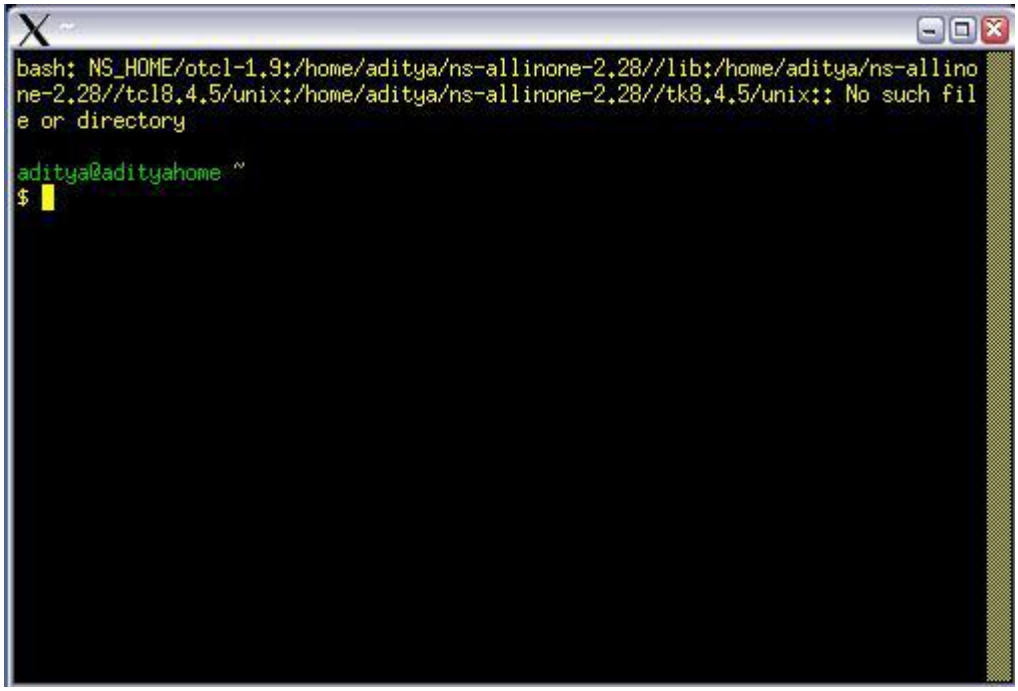
Catatan : kata yang dicetak tebal (bold) merupakan hasil pengembangan (penambahan atau perubahan) dari skrip simulasi asli versi Jorg Widmer.

LAMPIRAN B

TAMPILAN LAYAR SAAT SIMULASI

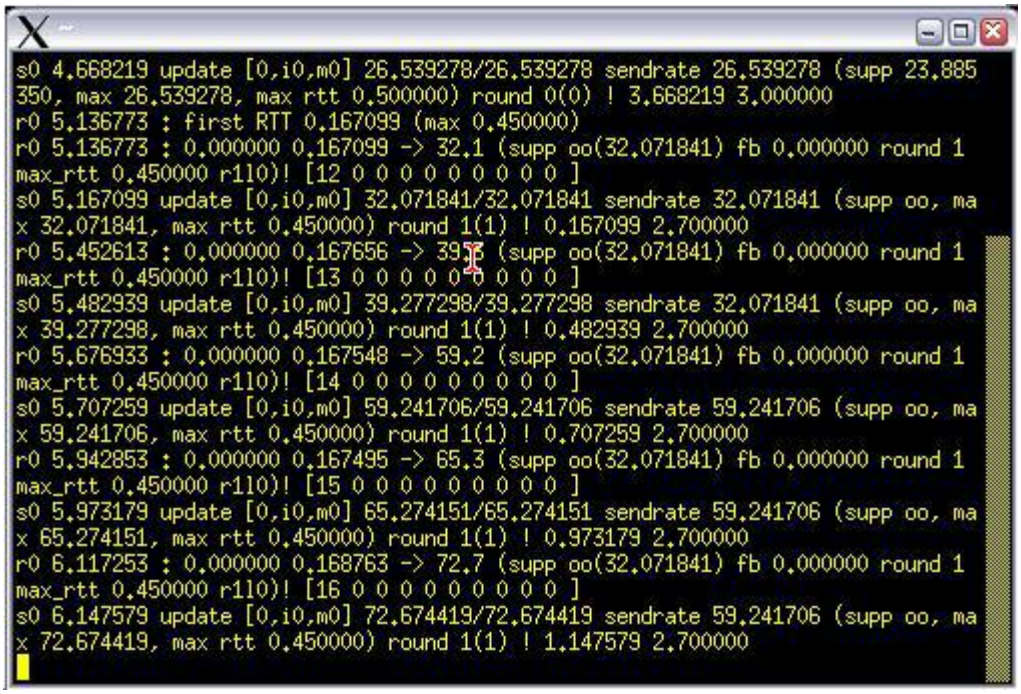


Gambar B-1
Tampilan Awal Layar Cygwin

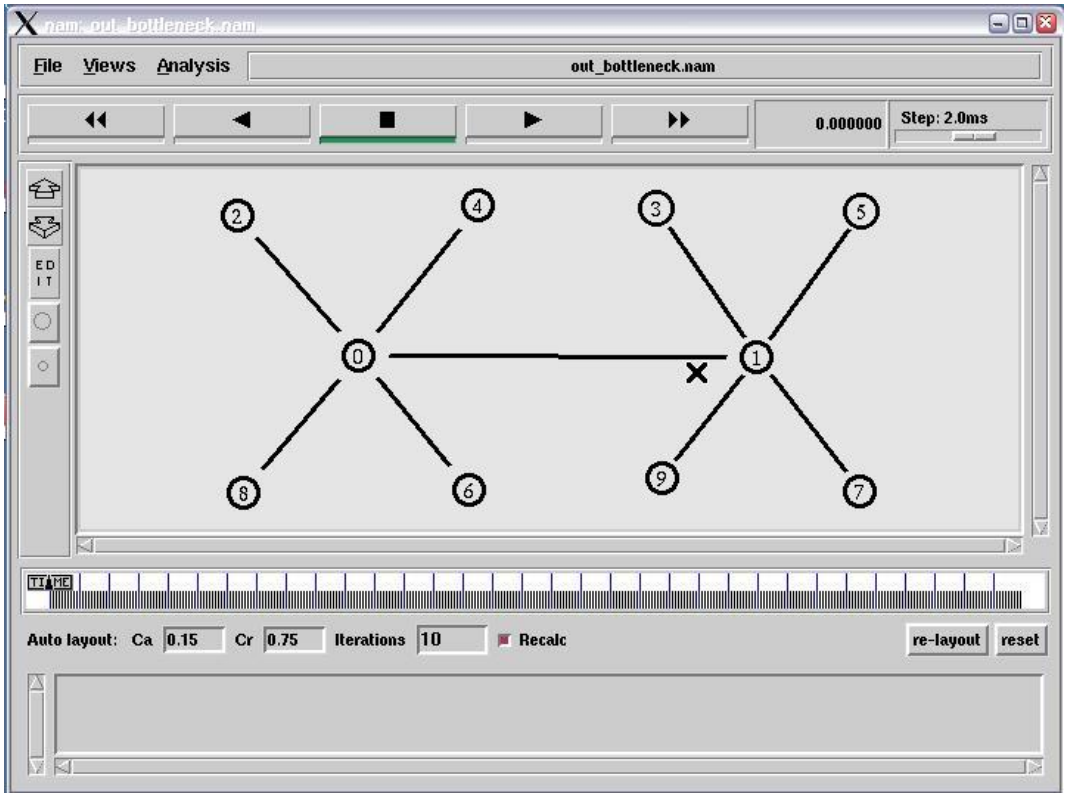


Gambar B-2
Tampilan Layar Cygwin Setelah Perintah startxwin.bat

Gambar B-3
Perintah untuk menjalankan simulasi TFMCC



Gambar B-4
Tampilan Layar saat simulasi berlangsung



Gambar B-5
Tampilan Layar NAM yang merupakan animasi hasil simulasi

LAMPIRAN C

SKRIP & PERINTAH PERL

1. SKRIP PERL UNTUK MENGHITUNG THROUGHPUT

```
$infile=$ARGV[0];
$tonode=$ARGV[1];
$granularity=$ARGV[2];

$sum=0;
$clock=0;

open (DATA,"<$infile")
||die "Can't open $infile $!";
while (<DATA>) {
@x=split(' ');

if ($x[1]-$clock <=$granularity)
{if ($x[0] eq 'r')
{if ($x[3] eq $tonode)
{if ($x[4] eq 'tcp')
# dapat diganti “if ($x[4] eq 'tfmcc_data')”  untuk throughput TFMCC
{$sum=$sum+$x[5];
}}}}
else
{$throughput=$sum/$granularity;
print STDOUT "$throughput\n";
$clock=$clock+$granularity;
$sum=0;
}}
$throughput=$sum/$granularity;
print STDOUT "$throughput\n";
$clock=$clock+$granularity;
$sum=0;
close DATA ;
exit(0);
```

(disimpan dengan nama throughput_tcp.pl atau throughput_tfmcc.pl)

PERINTAH PERL UNTUK MENGHITUNG THROUGHPUT

```
perl throughput_tcp.pl out_bottleneck.tr 3 1 > throughput_tcp_node_3
perl throughput_tfmcc.pl out_bottleneck.tr 3 1 > throughput_tfmcc_node_3
```

(perl “nama skrip perl untuk hitung throughput” “skrip hasil trace” “node tujuan” “interval sample” > “nama file hasil hitung”)

2. SKRIP PERL UNTUK MENGHITUNG JUMLAH PAKET KIRIM

```
$infile=$ARGV[0];
$fromnode=$ARGV[1];
$granularity=$ARGV[2];

$sum=0;
$clock=0;

open (DATA,"<$infile")
||die "Can't open $infile $!";
while (<DATA>) {
@x=split(' ');

if ($x[1]-$clock <=$granularity)
{if ($x[0] eq 'r')
{if ($x[2] eq $fromnode)
{if ($x[4] eq 'tcp')
# dapat diganti “if ($x[4] eq 'tfmcc_data')”  untuk paket kirim TFMCC
{$sum=$sum+($x[2]/$fromnode);
}}}}
else
{$pkirim=$sum;
print STDOUT "$pkirim\n";
$clock=$clock+$granularity;
$sum=0;
}}
$pkirim=$sum;
print STDOUT "$pkirim\n";
$clock=$clock+$granularity;
$sum=0;

close DATA ;
exit(0);
```

(disimpan dengan nama pkirim_tcp.pl atau pkirim_tfmcc.pl)

PERINTAH PERL UNTUK MENGHITUNG PAKET KIRIM

```
perl pkirim_tcp.pl out_bottleneck.tr 2 120 > pkirim_tcp_3
perl pkirim_tfmcc.pl out_bottleneck.tr 2 120 > pkirim_tfmcc
```

(perl “nama skrip perl untuk hitung paket kirim” “skrip hasil trace” “node asal” “interval sample” > “nama file hasil hitung”)

3. SKRIP PERL UNTUK MENGHITUNG JUMLAH PAKET TERIMA

```
$infile=$ARGV[0];
$tonode=$ARGV[1];
$granularity=$ARGV[2];

$sum=0;
$clock=0;

open (DATA,"<$infile")
||die "Can't open $infile $!";
while (<DATA>) {
@x=split(' ');

if ($x[1]-$clock <=$granularity)
{if ($x[0] eq 'r')
{if ($x[3] eq $tonode)
{if ($x[4] eq 'tcp')
# dapat diganti “if ($x[4] eq 'tfmcc_data')”  untuk paket terima TFMCC
{$sum=$sum+($x[3]/$tonode);
}}}}
else
{
$ptrima=$sum;
print STDOUT "$ptrima\n";
$clock=$clock+$granularity;
$sum=0;
}}
$ptrima=$sum;
print STDOUT "$ptrima\n";
$clock=$clock+$granularity;
$sum=0;

close DATA ;
exit(0);
```

(disimpan dengan nama ptrima_tcp.pl atau ptrima_tfmcc.pl)

PERINTAH PERL UNTUK MENGHITUNG PAKET TERIMA

```
perl ptrima_tcp.pl out_bottleneck.tr 3 120 > ptrima_tcp_3
perl ptrima_tfmcc.pl out_bottleneck.tr 3 120 > ptrima_tfmcc
```

(perl “nama skrip perl untuk hitung paket terima” “skrip hasil trace” “node tujuan” “interval sample” > “nama file hasil hitung”) [15]

LAMPIRAN D
CONTOH DAN PENJELASAN HASIL TRACE

CONTOH DAN PENJELASAN HASIL TRACE OUT_BOTTLENECK.TR

1. + 1 2 0 tfmcc_data 1000 ------- 0 2.1 -2147483648.0 -1 0
2. - 1 2 0 tfmcc_data 1000 ------- 0 2.1 -2147483648.0 -1 0
3. r 1.00508 2 0 tfmcc_data 1000 ------- 0 2.1 -2147483648.0 -1 0
4. + 1.00508 0 1 tfmcc_data 1000 ------- 0 2.1 -2147483648.0 -1 0
5. - 1.00508 0 1 tfmcc_data 1000 ------- 0 2.1 -2147483648.0 -1 0
6. r 1.033024 0 1 tfmcc_data 1000 ------- 0 2.1 -2147483648.0 -1 0
7. + 1.033024 1 3 tfmcc_data 1000 ------- 0 2.1 -2147483648.0 -1 0
8. - 1.033024 1 3 tfmcc_data 1000 ------- 0 2.1 -2147483648.0 -1 0
9. + 1.033024 1 5 tfmcc_data 1000 ------- 0 2.1 -2147483648.0 -1 0
10. - 1.033024 1 5 tfmcc_data 1000 ------- 0 2.1 -2147483648.0 -1 0
11. + 1.033024 1 7 tfmcc_data 1000 ------- 0 2.1 -2147483648.0 -1 0
12. - 1.033024 1 7 tfmcc_data 1000 ------- 0 2.1 -2147483648.0 -1 0
13. + 1.033024 1 9 tfmcc_data 1000 ------- 0 2.1 -2147483648.0 -1 0
14. - 1.033024 1 9 tfmcc_data 1000 ------- 0 2.1 -2147483648.0 -1 0
15. r 1.038104 1 3 tfmcc_data 1000 ------- 0 2.1 -2147483648.0 -1 0
16. r 1.038104 1 5 tfmcc_data 1000 ------- 0 2.1 -2147483648.0 -1 0
17. r 1.038104 1 7 tfmcc_data 1000 ------- 0 2.1 -2147483648.0 -1 0
18. r 1.038104 1 9 tfmcc_data 1000 ------- 0 2.1 -2147483648.0 -1 0

19. + 2.061961 4 0 tcp 1040 ------- 2 4.1 5.2 1 18
20. - 2.061961 4 0 tcp 1040 ------- 2 4.1 5.2 1 18
21. r 2.067044 4 0 tcp 1040 ------- 2 4.1 5.2 1 18
22. + 2.067044 0 1 tcp 1040 ------- 2 4.1 5.2 1 18
23. - 2.067044 0 1 tcp 1040 ------- 2 4.1 5.2 1 18
24. r 2.095306 0 1 tcp 1040 ------- 2 4.1 5.2 1 18
25. + 2.095306 1 5 tcp 1040 ------- 2 4.1 5.2 1 18
26. - 2.095306 1 5 tcp 1040 ------- 2 4.1 5.2 1 18
27. r 2.100389 1 5 tcp 1040 ------- 2 4.1 5.2 1 18
28. + 2.100389 5 1 ack 40 ------- 2 5.2 4.1 1 26
29. - 2.100389 5 1 ack 40 ------- 2 5.2 4.1 1 26
30. r 2.105393 5 1 ack 40 ------- 2 5.2 4.1 1 26
31. + 2.105393 1 0 ack 40 ------- 2 5.2 4.1 1 26
32. - 2.105393 1 0 ack 40 ------- 2 5.2 4.1 1 26
33. r 2.12571 1 0 ack 40 ------- 2 5.2 4.1 1 26
34. + 2.12571 0 4 ack 40 ------- 2 5.2 4.1 1 26
35. - 2.12571 0 4 ack 40 ------- 2 5.2 4.1 1 26
36. r 2.130714 0 4 ack 40 ------- 2 5.2 4.1 1 26

37. d 2.610824 0 1 tcp 1040 ------- 2 4.1 5.2 30 130

38. + 9.482586 9 1 tfmcc_ack 40 ------- 0 9.1 2.1 -1 1874
39. - 9.482586 9 1 tfmcc_ack 40 ------- 0 9.1 2.1 -1 1874
40. r 9.487589 9 1 tfmcc_ack 40 ------- 0 9.1 2.1 -1 1874
41. + 9.487589 1 0 tfmcc_ack 40 ------- 0 9.1 2.1 -1 1874
42. - 9.487589 1 0 tfmcc_ack 40 ------- 0 9.1 2.1 -1 1874
43. r 9.507907 1 0 tfmcc_ack 40 ------- 0 9.1 2.1 -1 1874
44. + 9.507907 0 2 tfmcc_ack 40 ------- 0 9.1 2.1 -1 1874
45. - 9.507907 0 2 tfmcc_ack 40 ------- 0 9.1 2.1 -1 1874
46. r 9.51291 0 2 tfmcc_ack 40 ------- 0 9.1 2.1 -1 1874

Penjelasan :
Kode antrean paket (+,-,r,d) / waktu simulasi (detik) / node pengirim / node tujuan / tipe data / packet size (bytes)
/ flags /
flow id / alamat sumber / alamat tujuan / packet sequence number / packet id

(+) = (enqueued / dimasukkan antrean),
(-) = (dequeued / dikeluarkan dari antrean),
(r) = (received / diterima oleh tujuan),
(d) = (dropped / dibuang)
prune = (menyebar),
ack = (acknowledge / feedback bahwa sudah diterima)

Nomor 1-18 Pengiriman paket TFMCC

Nomor satu menunjukkan, sebuah paket TFMCC 1000 *bytes* memasuki antrean pada detik pertama. Paket ini dikirim oleh *node* 2 menuju *node* 0. Pada nomor 2, paket TFMCC tersebut dinyatakan ditelah sampai di *node* 0. Kemudian disampaikan ke *node* 1, dan akhirnya disebarkan ke *node* 3, 5, 7, dan 9.

Nomor 19-36 Pengiriman dan *feedback* Paket TCP

Nomor 19 menunjukkan sebuah paket TCP 1040 *bytes* dikirim oleh *node* 4 menuju *node* 0 pada detik 2,61. Packet ini mempunyai *packet ID* 18 hingga tiba di *node* tujuan yaitu *node* 5. Kemudian *node* 5 akan mengirimkan ACK ke *node* pengirim dengan *packet ID* yang lain yaitu 26.

Nomor 37 Contoh kode antrean paket yang *drop* atau dibuang.

Nomor 38 Pengiriman *feedback* TFMCC

Feedback dikirimkan oleh *node* 9 menuju pengirim yaitu *node* 2, dengan *pakcet ID* 1874.

LAMPIRAN E
DATA STATISTIK