3.Lab 3

Sunday, February 14, 2021 2:01 PM

TCL: tool command language

.tr
Trace, transmission file
Transmission details (time etc.)

.nam
Animation file to visualize your network

VMWare Workstation
Open source, needs open source Operating System

Ns-2 (network simulator 2)

Command: ns Wired_First.tcl
>nam out.nam
//shows output

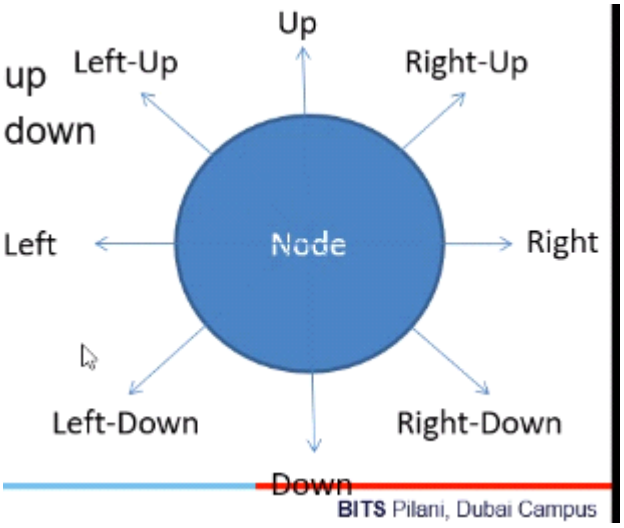
UDP: no acknowledgement
TCP: acknowledgement
Hybrid traffic: UDP + TCP

Delay introduced due to link: propagation delay
0 - 2: 2mb(capacity), 10ms(propagation delay)

Source	Destination	
TCP	TCPSink	Generates acknowledement
UDP	NULL	No ack.

Different classes
Ns-2 is OO-simulator
TCP, TCPSink are classes
Transport layer
TCP/IP model -> application layer after transport layer
You cannot use FTP with UDP, only with TCP

Orientations




```
set ns[new Simulator]
#create simulator object called ns
#Simulator is a class
#set creates variable
#ns is variable name
#new is a keyword to create object
#must for every program
#ns is object of simulator
#$ns means reference of the object
#equivalent to & in C, C++

#color is a method from Simulator
#1, 2 are flow IDs
#blue assigned to 1, red assigned to 2
$ns color 1 Blue
$ns color 2 Red

#the simulation results will be stored here
#file1 is a variable - file pointer
#open to create and open your file out.tr
#w --write mode (transmission details go here)
set file1 [open out.tr w]
#all activity goes into file1 --> out.tr
#trace-all captures all transmission details
#needs Simulator object
$ns trace-all $file1
#NAM trace file
set file2 [open out.nam w]
#capturing animation
$ns namtrace-all $file2


#creating nodes
#physical layer
#addressing is for network layer, n0,n1... are similar to IP addresses
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
```

```
set n4 [$ns node]
set n5 [$ns node]


#create links between nodes
#duplex(bidirectional) and simplex(unidirectional) links
#queues on a link to synchronize speeds --DropTail
#every link between sender and receiver needs a queue
#size of queue is initially infinite
#object link-type node1 node2 transmission_speed prop_delay queue_type
#physical layer
#network layer (routing)
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns simplex-link $n2 $n3 0.3Mb 100ms DropTail
$ns simplex-link $n3 $n2 0.3Mb 100ms DropTail
#these two links can be converted to duplex link
$ns duplex-link $n3 $n4 0.5Mb 40ms DropTail
$ns duplex-link $n3 $n5 0.5Mb 30ms DropTail


#every node and link has an orientation
$ns duplex-link-op $n0 $n2 orient right-down
#orient n2 with respect to n0
$ns duplex-link-op $n1 $n2 orient right-up
#orient n1 with respect to n0
$ns simplex-link-op $n2 $n3 orient right
$ns simplex-link-op $n3 $n2 orient left
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $n3 $n5 orient right-down
#one orientation necessary for every link



#you can limit your queue size, default size is infinite
#queue-limit
#data link layer
$ns queue-limit $n2 $n3 40
#if more than 40 packets, drop packet


#transport layer
#creating object of TCP class (tcp)
#Agent is superclass
set tcp [new Agent/TCP]
#set n0 as tcp source
#connecting tcp object to tcp source (attach-agent)
$ns attach-agent $n0 $tcp
#creating object of TCPSink (for TCP destination)
set sink [new Agent/TCPSink]
$ns attach-agent $n4 $sink
#logical connection from source transport layer tcp to destination tcpsink
$ns connect $tcp $sink
#set flow id for the tcp connection
$tcp set fid_ 1
#create packet size with 552bytes
$tcp set packetSize_ 552
```

```
#application layer for TCP is FTP
#application layer object created
set ftp [new Application/FTP]
#application is super class
$ftp attach-agent $tcp


#connection for UDP
#source: n1, destination: n5
#creating object for UDP class
set udp [new Agent/UDP]
#connect to the source node n1
$ns attach-agent $n1 $udp
#object of null class (for  UDP destination)
set null [new Agent/Null]
$ns attach-agent $n5 $null
#connect UDP and null
$ns connect $udp $null
#flow id 2 -- red color
$udp set fid_ 2


#cbr is application layer for UDP
#constant bit rate
set cbr [new Application/Traffic/CBR]
#attach cbr to UDP
$cbr attach-agent $udp
#1000 bytes for UDP traffic
$cbr set packet_size_ 1000


#user accesses network through application layer
#traffic/event needs to be scheduled/generated
#UDP traffic starts at 0.1
$ns at 0.1 "$cbr start"
#TCP traffic at 1.0
$ns at 1.0 "$ftp start"
$ns at 624.0 "$ftp stop"
$ns at 624.5 "$cbr stop"


#calling finish procedure at 625s
#finish is part of program so no need for object to call it
$ns at 625.0 “finish”


#Run the simulation
#equivalent to void main() in C
#script runs
$ns run


#proc keyword to define procedure
#{} parameters inside
proc finish {} {
    #use ns file1 file2 used globally
    global ns file1 file2
```

```
    #flush-trace will deallocate memory
    #applied to ns which is the main object, all objects connected to it
    $ns flush-trace
    #files closed
    close $file1
    close $file2
    exit 0 }

#run script ns filename.tcl
#run animation nam filename.nam
```