Studying TCP's Congestion Window using NS

- ## How to obtain TCP's CWND value

  - The most important value that determine the behavior of TCP is the **congestion window size** or traditionally abreviated as **CWND**

  - In **NS**, every **TCP-type** class (**Agent/TCP/Tahoe**, (**Agent/TCP/Reno**, etc) has a variable named

    ```
    cwnd_
    ```

    that contains the **congestion window size** of the TCP module

  - Recall that we can use the `set` command to return a value

  - Hence, the following command will retrieve the **congestion window size** of a TCP module:

    ```
    set  tcp1  [new  Agent/TCP/Reno]

    set  cwnd1  [ $tcp1  set  cwnd_ ]      // read variable "cwnd_"
    ```

- ## How to obtain TCP's CWND value *PERIODICALLY*

  - Now that we know how to read the **congestion window size** of a TCP module **once**, it is easy to make the **NS** simulation system repeatedly read the value (say, after every 0.1 sec of simulation time).

  - All we need to do is to **schedule a read operation *repeatedly***

  - We have seen an example of **self-scheduling** behavior in the "2 person talking example" (click here)

  - We can use a similar **self-scheduling** procedure to obtain the value of **CWND** repeated.

  - **Example:** (requires that the **Simulator** object variable be named **$ns**)

    ```
    proc plotWindow {tcpSource outfile} {
       global ns

       set now [$ns now]
       set cwnd [$tcpSource set cwnd_]
    ```

```
#  Print TIME CWND   for  gnuplot to plot progressing on CWND
   puts  $outfile  "$now $cwnd"

   $ns at [expr $now+0.1] "plotWindow  $tcpSource  $outfile"
}
```

1. The procedure **plotWindow** takes a paramter **tcpSource** which is a TCP agent

   So you can use the procedure to plot the **CWND** from any number of TCP flows.

2. The procedure **plotWindow** takes an output file ID **outfile**

   You should first open an output file (or use "stdout") in the main program

- **Examining progressing of CWND in TCP (Reno)**
  - Here is the previous example (click here) which additional code to obtain the **congestion window size** of the TCP module **$tcp1**:

    (New code is colored as **magenta**)

```
#Make a NS simulator
set ns [new Simulator]

# Define a 'finish' procedure
proc finish {} {
   exit 0
}

# Create the nodes:
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 the links:
$ns duplex-link $n0 $n2   2Mb  10ms DropTail
$ns duplex-link $n1 $n2   2Mb  10ms DropTail
$ns duplex-link $n2 $n3 0.3Mb 200ms DropTail
$ns duplex-link $n3 $n4 0.5Mb  40ms DropTail
$ns duplex-link $n3 $n5 0.5Mb  30ms DropTail

# Add a TCP sending module to node n0
set tcp1 [new Agent/TCP/Reno]
$ns attach-agent $n0 $tcp1

# Add a TCP receiving module to node n4
set sink1 [new Agent/TCPSink]
$ns attach-agent $n4 $sink1

# Direct traffic from "tcp1" to "sink1"
$ns connect $tcp1 $sink1
```

```
# Setup a FTP traffic generator on "tcp1"
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ftp1 set type_ FTP               (no necessary)

# Schedule start/stop times
$ns at 0.1   "$ftp1 start"
$ns at 100.0 "$ftp1 stop"

# Set simulation end time
$ns at 125.0 "finish"             (Will invoke "exit 0")


##################################################
## Obtain CWND from TCP agent
##################################################

proc plotWindow {tcpSource outfile} {
   global ns

   set now [$ns now]
   set cwnd [$tcpSource set cwnd_]

###Print TIME CWND   for  gnuplot to plot progressing on CWND
   puts  $outfile  "$now $cwnd"

   $ns at [expr $now+0.1] "plotWindow $tcpSource  $outfile"
}

$ns  at  0.0  "plotWindow $tcp1  stdout"   // Start the probe !!


# Run simulation !!!!
$ns run
```

- **Example Program:** (Demo above code)

  - This NS Prog prints the (time, cwnd) to the terminal: click here
  - This NS Prog prints the (time, cwnd) to the output file "WinFile": click here

To run the program, use the command:

ns Reno2.tcl

To plot the window progressing from "winfile", do:

- **UNIX>> gnuplot**
- **gnuplot>> plot "WinFile" using 1:2 title "Flow 1" with lines 1**

- **NOTE:**

In case you wonder why the CWND plot look so different, it's because the setting of some parameters.

Add the following statements to the simulation to get the one I used in class:

```
# ########################################################
# Set Queue Size of link (n2-n3) to 10 (default is 50 ?)
# ########################################################
$ns queue-limit $n2 $n3 10


# ########################################################
# TCP parameters:
# ########################################################
$tcp1 set window_ 8000
$tcp1 set packetSize_ 552
```

- This NS Prog will draw the CWND: click here

- **Postscript: Analyzing multiple TCP flows**
  - The easiest way to analyze the behavior of multiple TCP is to open one file to store the progression of one TCP agent's variable values.
  - **Example: 2 TCP Agents**

```
set tcp1 [new Agent/TCP/Reno]
...
set tcp2 [new Agent/TCP/Reno]
...


set outfile1  [open  "WinFile1"  w]
set outfile2  [open  "WinFile2"  w]


$ns  at  0.0  "plotWindow $tcp1  $outfile1"

$ns  at  0.0  "plotWindow $tcp2  $outfile2"
```

Plot data of **TCP 1** will be store in file "WinFile1"

Plot data of **TCP 2** will be store in file "WinFile2"