DelayBox: Per-Flow Delay and Loss

DelayBox is an ns node that should be placed in between the source and destination nodes. With Delaybox, packets from a TCP flow can be delayed, dropped, and/or forced through a bottleneck link before being passed on to the next node. A distribution can be used to specify delay, loss, and/or bottleneck link speed for a source - destination pair. Each flow between that source - destination pair draws from the distribution to determine its characteristics. Delays in DelayBox are per-flow, rather than per-packet. Since DelayBox distinguishes between flows, the `fid_` variable (flow identifier) should be set for each flow in the simulation. DelayBox can be used with both Tcp and FullTcp agents.

Implementation Details

DelayBox maintains two tables: a rule table and a flow table. Entries in the rule table are added by the user in the OTcl simulation script and give an outline of how flows from a source to a destination should be treated. The fields are source, destination, delay Random Variable (in ms), loss rate Random Variable (in fraction of packets dropped), and bottleneck link speed Random Variable (in Mbps). The bottleneck link speed field is optional. Entries in the flow table are created internally and specify exactly how each flow should be handled. Its values are obtained by sampling from the distributions given in the rule table. The fields are source, destination, flow ID, delay, loss, and bottleneck link speed (if applicable). Full-TCP flows are defined as beginning at the receipt of the first SYN of a new flow ID and ending after the sending of the first FIN. Packets after the first FIN are forwarded immediately (i.e., they are neither delayed nor dropped by DelayBox). For TcpAgent, flows are defined as beginning at the receipt of the first 40 byte packet of a new flow ID. Since there are no FIN packets in TcpAgent, TcpAgent flows are never considered finished nor are they removed from the flow table.

DelayBox also maintains a set of queues to handle delaying packets. There is one queue per entry in the flow table. These queues are implemented as delta queues, in that the time to transfer the packet is kept only for the head packet. All other packets are stored with the difference between the time they should be transferred and the time the previous packet should be transferred. The actual time the previous packet should be transferred is stored in the variable `deltasum_`, named so because it is the sum of all delta values in the queue (including the head packet's transfer time). If the bottleneck link speed has been specified for the flow, a processing delay is computed for each packet by dividing the size of the packet by the flow's bottleneck link speed.

When a packet is received, its transfer time (current time + delay) is calculated. (This transfer time is the time that the first bit of the packet will begin transfer. Packets that wait in the queue behind this packet must be delayed by the amount of time to transfer all bits of the packet over the bottleneck link.) There are two scenarios to consider in deciding how to set the packet's delta:

1. If the packet is due to be transferred before the last bit of the last packet in the queue, its delta (the time between transferring the previous packet and transferring this packet) is set to the previous packet's processing delay. This packet has to queue behind the previous packet, but will be ready to be transmitted as soon as the previous packet has completed its transfer.

2. If the packet is due to be transferred after the last bit of the last packet in the queue, its delta is difference between this packet's transfer time and the previous packet's transfer time.

If the current packet is the only packet in the queue, DelayBox schedules a timer for the receipt of the packet. When this timer expires, DelayBox will pass the packet on to the standard packet forwarder for processing. Once a packet has been passed up, DelayBox will look for the next packet in the queue to be processed and schedule a timer for its transfer. All packets, both data and ACKs, are delayed in this manner.

Packets that should be dropped are neither queued nor passed on. All packets in a queue are from the same connection and are delayed the same amount (except for delays due to packet size) and are dropped with the same probability. **Note:** Drops at DelayBox are not recorded in the trace-queue file.

Example

More examples are available in the `tcl/ex/delaybox/` directory of the ns source code. The validation script `test-suite-delaybox` is in `tcl/test/` and can be run with the command `test-all-delaybox` from that directory.

```
# test-delaybox.tcl - NS file transfer with DelayBox

# setup ns
remove-all-packet-headers;            # removes all packet headers
add-packet-header IP TCP;             # adds TCP/IP headers
set ns [new Simulator];               # instantiate the simulator

global defaultRNG
$defaultRNG seed 999

# create nodes
set n_src [$ns node]
set db(0) [$ns DelayBox]
set db(1) [$ns DelayBox]
set n_sink [$ns node]

# setup links
$ns duplex-link $db(0) $db(1) 100Mb 1ms DropTail
$ns duplex-link $n_src $db(0) 100Mb 1ms DropTail
$ns duplex-link $n_sink $db(1) 100Mb 1ms DropTail

set src [new Agent/TCP/FullTcp]
set sink [new Agent/TCP/FullTcp]
$src set fid_ 1
$sink set fid_ 1

# attach agents to nodes
$ns attach-agent $n_src $src
$ns attach-agent $n_sink $sink

# make the connection
$ns connect $src $sink
$sink listen
```

```
# create random variables
set recvr_delay [new RandomVariable/Uniform];     # delay 1-20 ms
$recvr_delay set min_ 1
$recvr_delay set max_ 20
set sender_delay [new RandomVariable/Uniform];    # delay 20-100 ms
$sender_delay set min_ 20
$sender_delay set max_ 100
set recvr_bw [new RandomVariable/Constant];       # bw 100 Mbps
$recvr_bw set val_ 100
set sender_bw [new RandomVariable/Uniform];       # bw 1-20 Mbps
$sender_bw set min_ 1
$sender_bw set max_ 20
set loss_rate [new RandomVariable/Uniform];       # loss 0-1% loss
$loss_rate set min_ 0
$loss_rate set max_ 0.01

# setup rules for DelayBoxes
$db(0) add-rule [$n_src id] [$n_sink id] $recvr_delay $loss_rate $recvr_bw
$db(1) add-rule [$n_src id] [$n_sink id] $sender_delay $loss_rate $sender_bw

# output delays to files
$db(0) set-delay-file "db0.out"
$db(1) set-delay-file "db1.out"

# schedule traffic
$ns at 0.5 "$src advance 10000"
$ns at 1000.0 "$db(0) close-delay-file; $db(1) close-delay-file; exit 0"

# start the simulation
$ns run
```

Commands at a Glance

The following commands on the DelayBox class can be accessed from OTcl:

`[$ns DelayBox]`
Creates a new DelayBox node.

`$delaybox add-rule <srcNodeID> <dstNodeID> <delayRV> [<lossRV>] [<linkSpeedRV>]`
Add a rule to the rule table, specifying delay, loss rate, and bottleneck link speed RandomVariables for packets flowing from `srcNode` to `dstNode`. Delay is required, but loss rate and link speed are optional.

`$delaybox list-rules`
List all rules in the rule table

`$delaybox list-flows`
List all flows in the flow table

`$delaybox set-asymmetric`
Specifies that the delay should be only on the data path rather than applied to both the data and ACK paths

`$delaybox set-delay-file <filename>`
Output delays for each flow to `filename`. Format: `srcNode dstNode fid delay`

`$delaybox close-delay-file`
Closes the file where delays are written

`$delaybox set-debug <int>`
Set the debugging level

- 1: Output when packets are dropped at DelayBox
- 2: Level 1 +
  Contents of the queue at each queue operation