The Greatest Network Simulation

Hongjian L, Yamei O, Samuel R, Jan VB, Junlin Z

Preliminary Implementation Of All Time

Presentation Outline

- Background
- Component Status
- Initial Results

Background

- SimPy
  - Actors
    - Flow, Host, Link, Router
  - Packets
    - DataPacket, RouterPacket
  - Events
    - FlowStart, LinkAvailable, PacketRecipt, PacketTimeOut, RoutingTableOutdated

Links

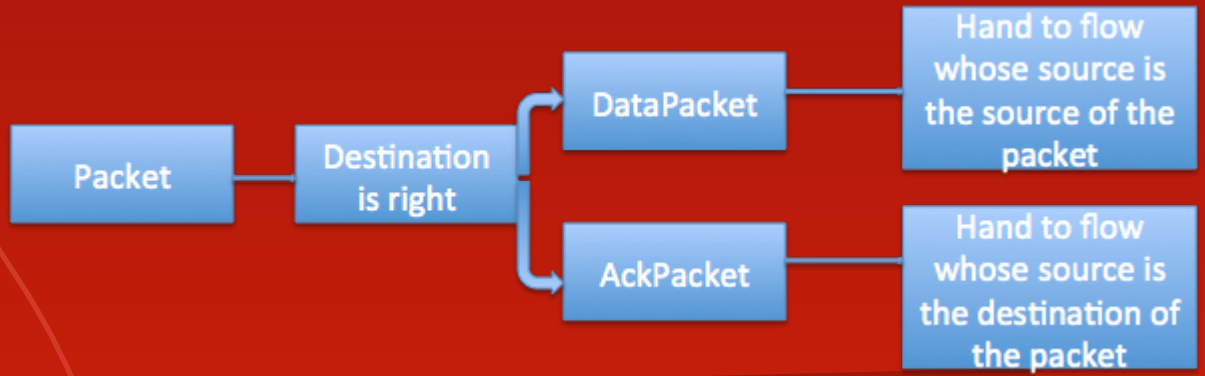
- Split each link into a & b (Full Duplex)
- Add packet to buffer (buffer drops if full)
- Schedule “Packet Receipt” events
- Transmit new packet when link available

Host

- Host()
  - address(Its IP Address)
  - flows(List of flows on this host)
  - link(The link connected to this host)
- send()
- react_to_packet_receipt()

react_to_packet_receipt

- If the destination of the packet is not the host's IP address, then drop it.
- If the destination is right and the packet is a DataPacket, then hand it to the flow whose source is the source of the packet.
- If the destination is tight and the packet is an AckPacket, then hand it to the flow whose source is the destination of the packet.



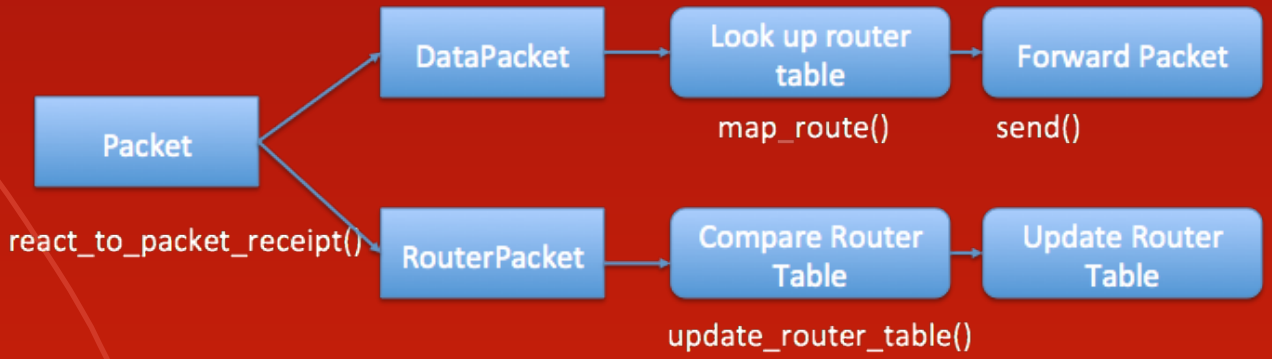
Router

- Router():
  - address(Unique IP address)
  - links (list of links connected to router)
  - table(Key-->destination, Val-->(metric, next hop))
  - default_gateway(link[0])
- react_to_packet_receipt()
- react_to_router_table_outdated()

react_to_packet_receipt

Tell the packet is a RouterPacket or DataPacket after receiving a packet from links.

- DataPacket: Forward to next link according the current routing table.
- RouterPacket:
  - Compare router table from RouterPacket and the current router.
  - Update router table if there is a shorter path.



react_to_router_table_outdated

- Set a timer to send RouterPacket periodically.
- Generate RouterPacket (source,timestamp, routertable).
- Send to all links connected to this router.
- Metric is based on hops.

Result

•Testcase1:

At 1190 R1 received RouterPacket from R3

{'1': (3, 'R2'), '0': (1, '0')}

At 1190 R2 received RouterPacket from R4

{'1': (2, 'R4'), '0': (2, 'R1')}

At 1190 R4 received RouterPacket from R3

{'1': (1, '1'), '0': (3, 'R2')}

At 1190 R3 received RouterPacket from R4

{'1': (2, 'R4'), '0': (2, 'R1')}

Result

•Testcase2:

At 1190 R1 received RouterPacket from R2

{'S3': (3, 'R2'), 'S2': (1, 'S2'), 'S1': (1, 'S1'), 'T2': (2, 'R2'), 'T3': (4, 'R2'), 'T1': (4, 'R2')}

At 1190 R2 received RouterPacket from R3

{'S3': (2, 'R3'), 'S2': (2, 'R1'), 'S1': (2, 'R1'), 'T2': (1, 'T2'), 'T3': (3, 'R3'), 'T1': (3, 'R3')}

At 1190 R3 received RouterPacket from R4

{'S3': (1, 'S3'), 'S2': (3, 'R2'), 'S1': (3, 'R2'), 'T2': (2, 'R2'), 'T3': (2, 'R4'), 'T1': (2, 'R4')}

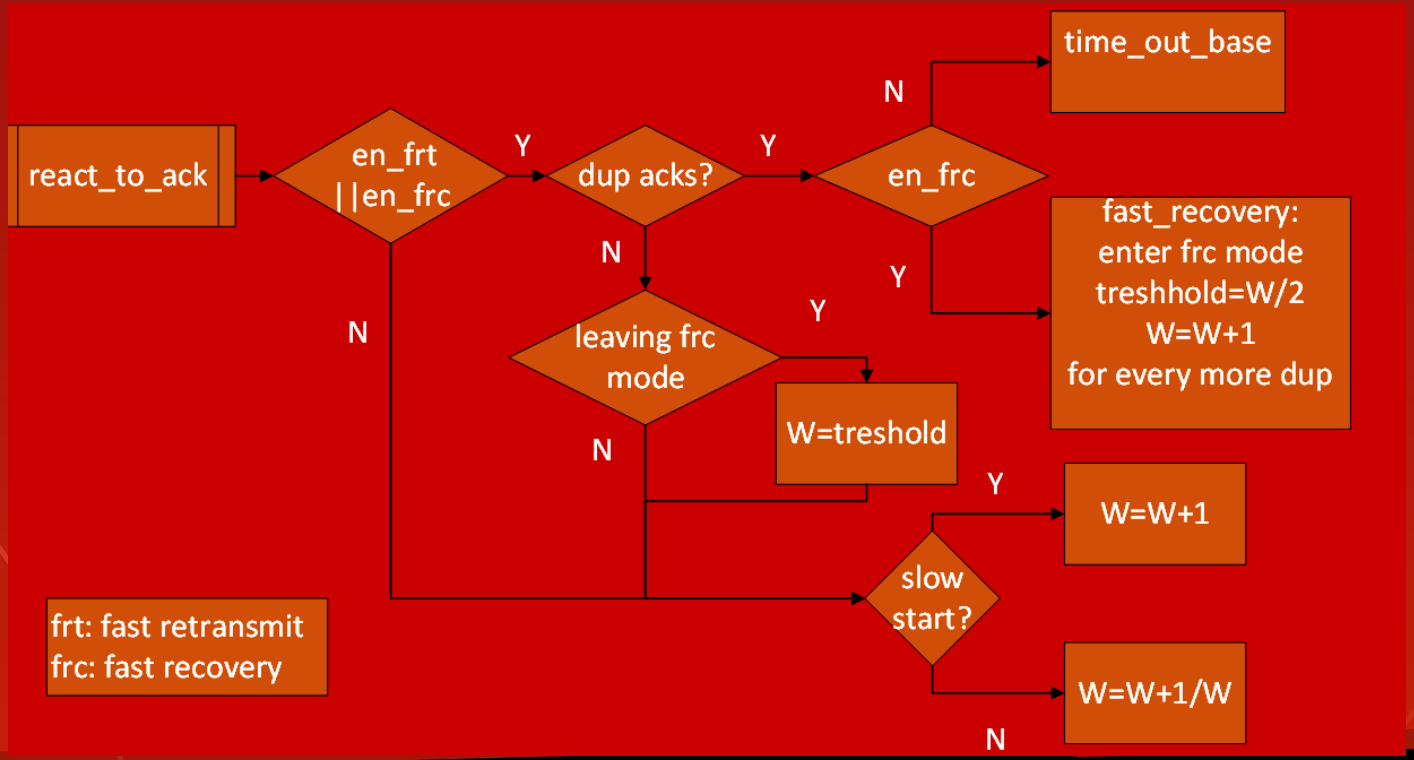
At 1190 R4 received RouterPacket from R3

{'S3': (2, 'R3'), 'S2': (4, 'R3'), 'S1': (4, 'R3'), 'T2': (3, 'R3'), 'T3': (1, 'T3'), 'T1': (1, 'T1')}

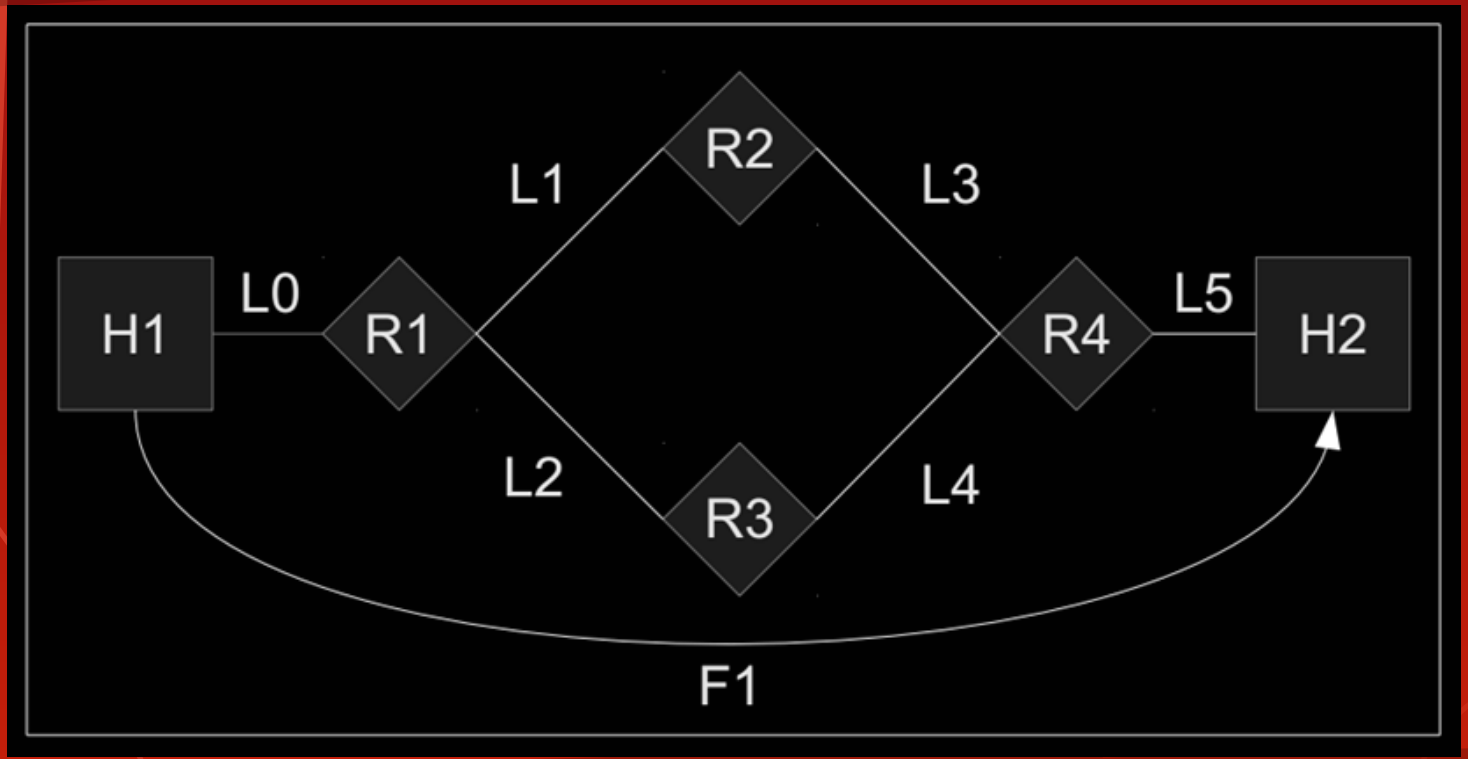
Transport Layer Algorithm

- react_to_flow_start
- react_to_ack
- react_to_time_out
- react_to_time_out_base
- send_new_packets

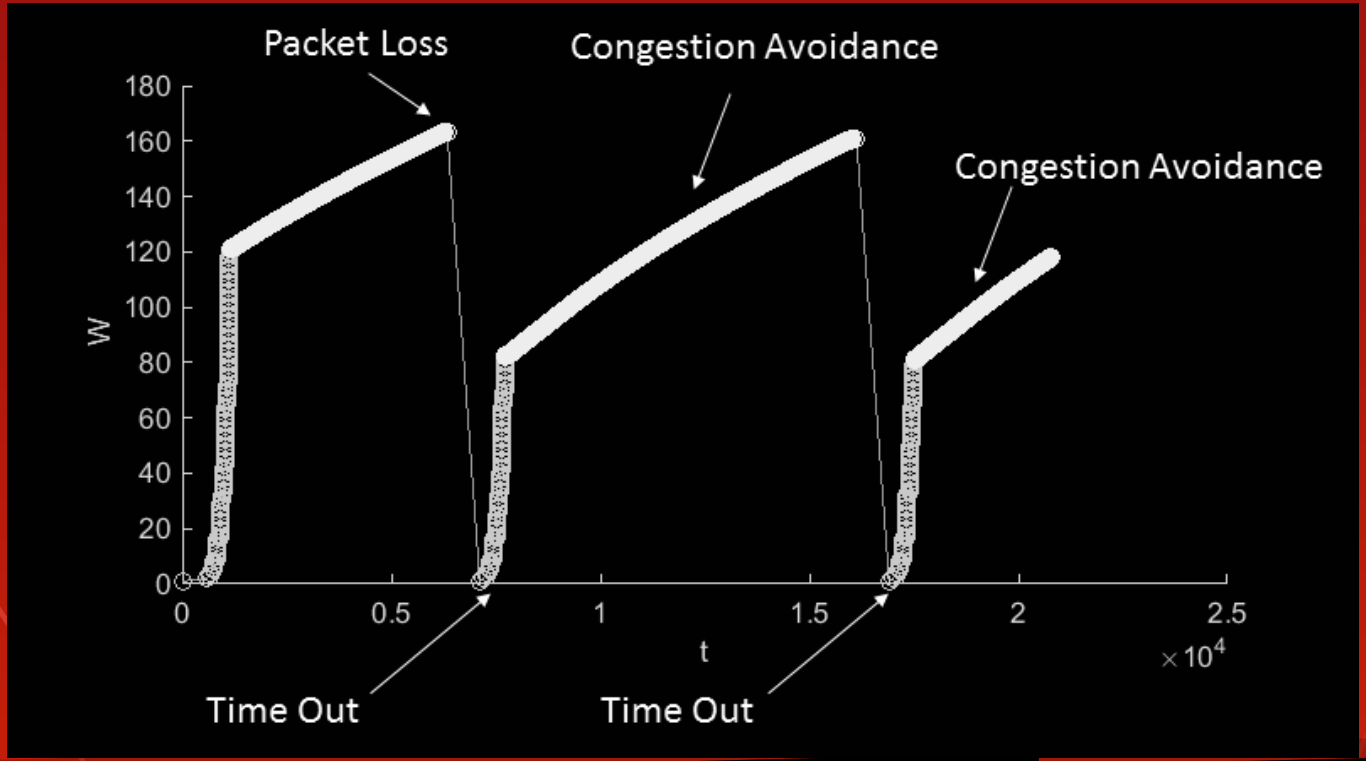
react_to_ack


react_to_ack
en_frt
||en_frc
Y
dup acks?
Y
en_frc
N
time_out_base
Y
fast_recovery:
enter frc mode
treshhold=W/2
W=W+1
for every more dup
N
leaving frc
mode
Y
W=treshold
N
N
Y
W=W+1
slow
start?
W=W+1/W
N
frt: fast retransmit
frc: fast recovery

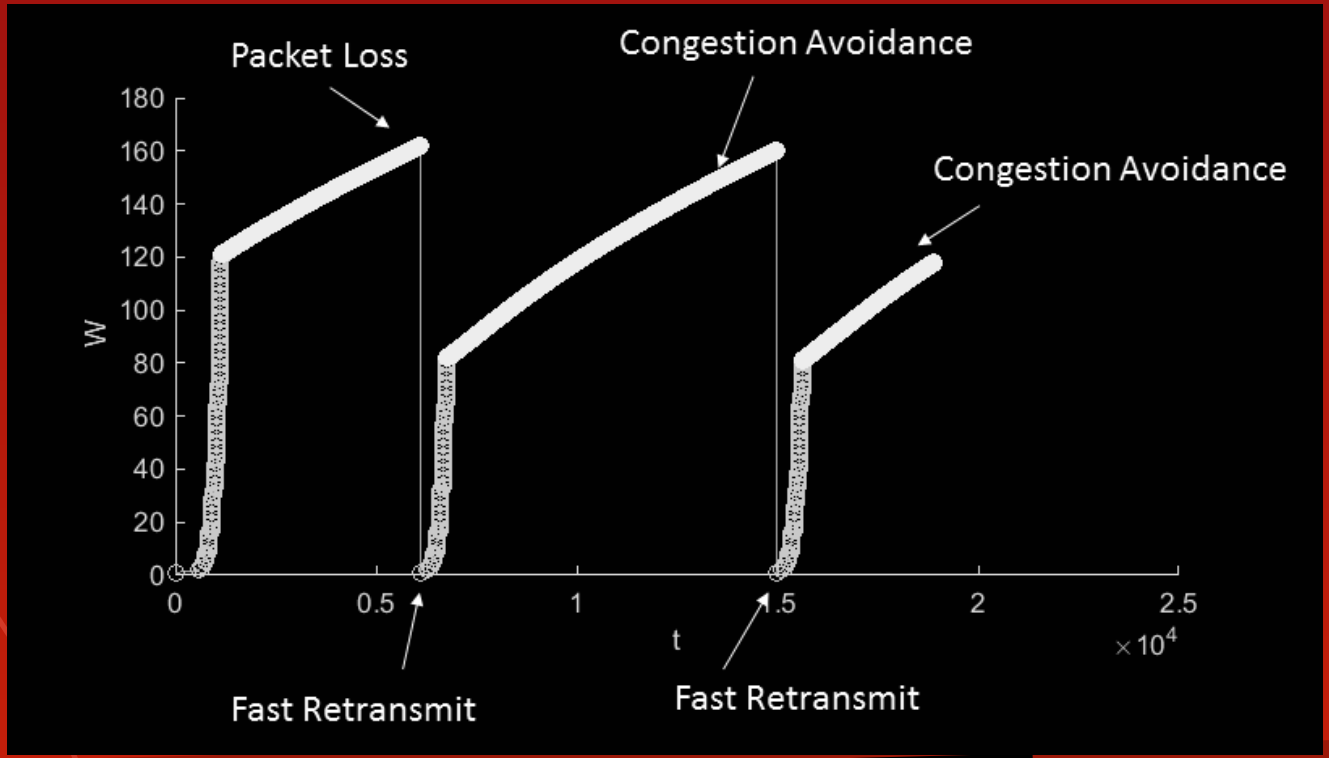
Test Case 1



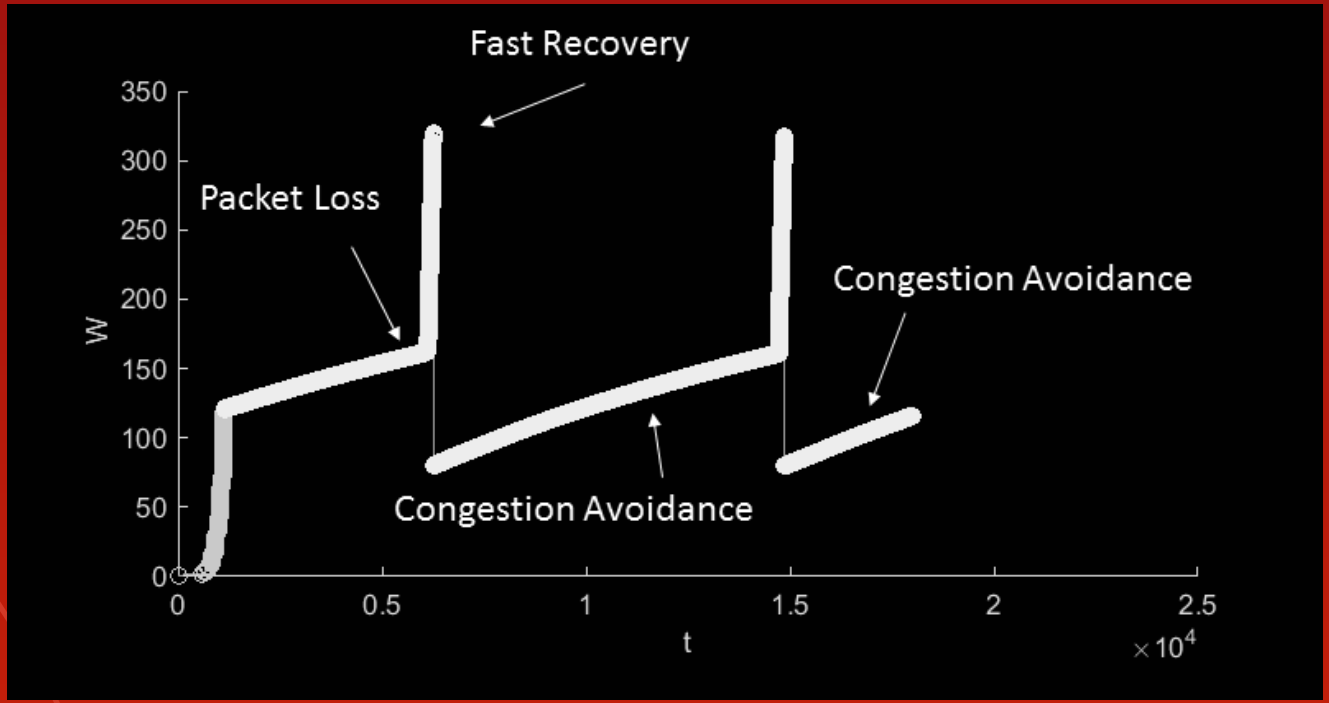
Without frt or frc



Fast Retransmit

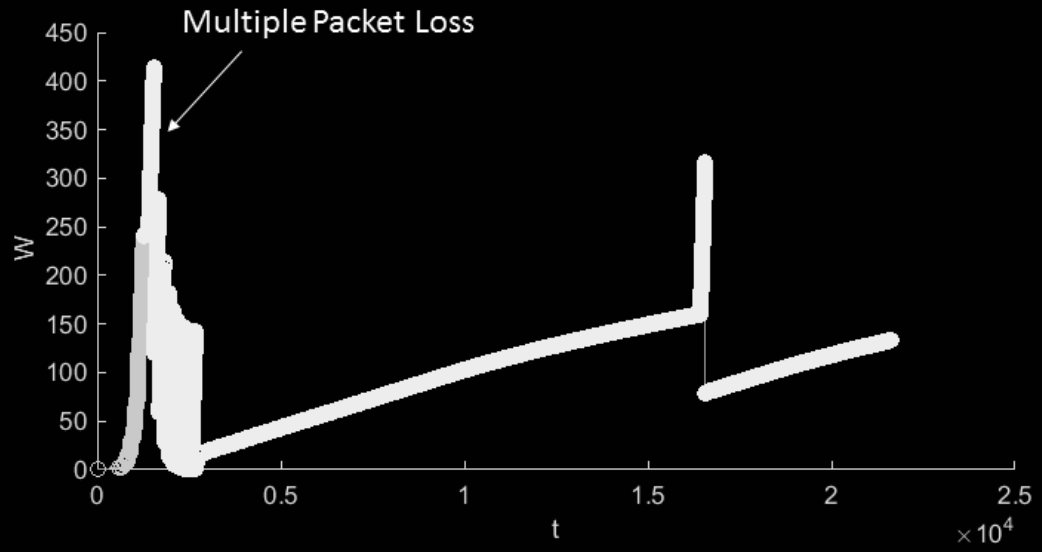


Fast Recovery

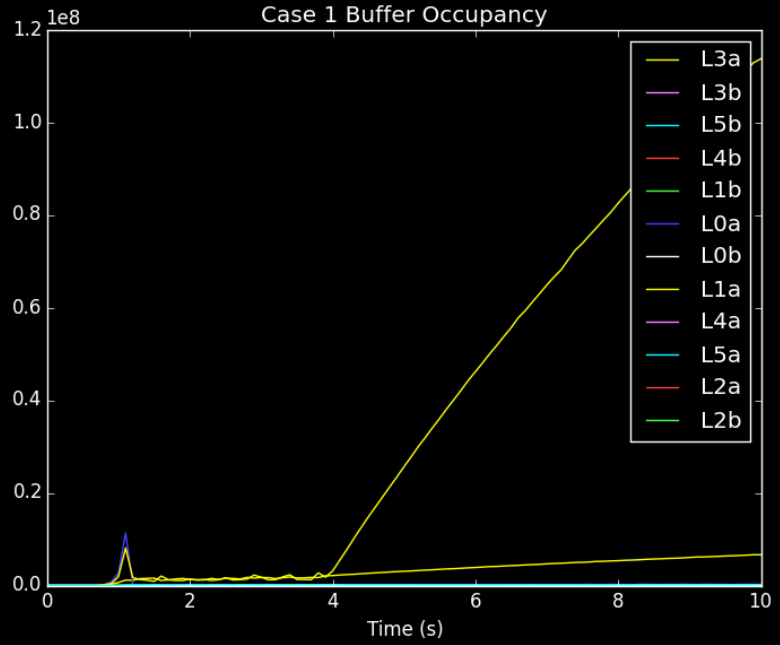


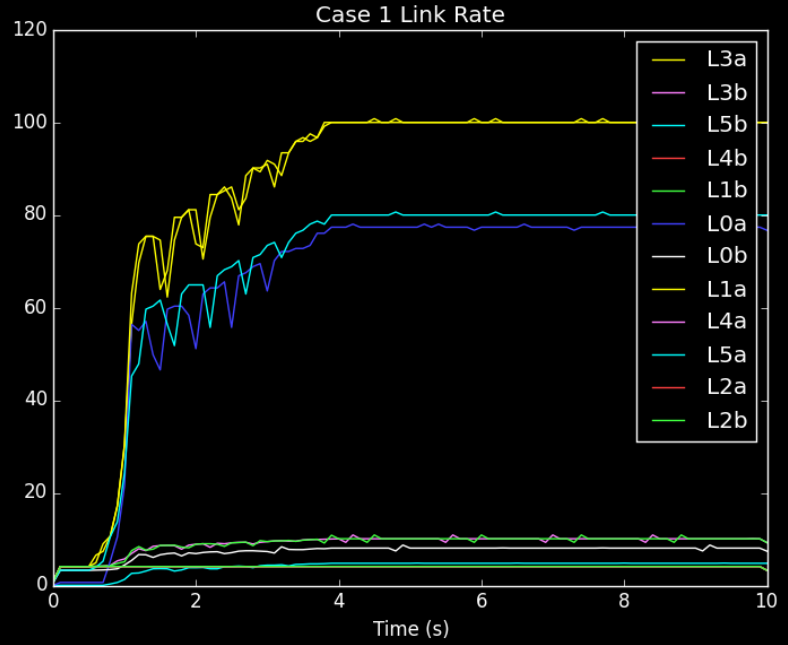
Problem: Multiple Packet Loss

- Multiple Packet Loss: Bad Behavior



Test Cases


Case 1 Buffer Occupancy
1e8
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
Time (s)
L3a
L3b
L5b
L4b
L1b
L0a
L0b
L1a
L4a
L5a
L2a
L2b


Case 1 Link Rate
120
100
80
60
40
20
0
0
2
4
6
8
10
Time (s)
L3a
L3b
L5b
L4b
L1b
L0a
L0b
L1a
L4a
L5a
L2a
L2b

Questions