Computer Networks (CO300)

Project Documentation

Project Topic: Validate the ns-3 implementation of BIC TCP

Submitted By:
Namrata Ladda 16CO121
Mehnaz Yunus 16CO124
Sharanya Kamath 16CO140

Submitted to:
Prof. Mohit P. Tahiliani
Assistant Professor
CSE Department, NITK

1. Abstract

Binary Increase Congestion control (BIC) is a precursor to CUBIC and is targeted for Long Fat Networks. It uses a binary search algorithm to find the optimal value of the the congestion window (cwnd). In this project, the aim is to validate ns-3 BIC implementation by comparing the results obtained from it to those obtained by simulating Linux BIC.

2. Aim

This project aims to validate the ns-3 implementation of TCP BIC. This is to be done by comparison of the congestion window and queue graphs after running both the ns-3 and Linux stacks for TCP BIC on the end nodes used in the simulation. A dumbbell topology with five senders and five receivers has been used here.

3. Theory

a. Direct Code Execution(DCE)

Direct Code Execution (DCE) is a framework for ns-3 that provides facilities to execute, within ns-3, existing implementations of userspace and kernel space network protocols or applications without source code changes. Here, DCE is used to run the actual Linux stack on the nodes in the simulation.

b. TCP BIC

BIC TCP is used for high-speed networks. It is used to reduce the time taken to overcome the difference in congestion window due to packet loss. It supports scalability, RTT unfairness and TCP friendliness.

BIC consists of two parts Binary increase and Additive increase:

Binary Search Increase: If W_{min} is the size of the congestion window just after last fast recovery and W_{max} is the maximum size of congestion window, mostly the size of the window just before the last fast recovery, the algorithm repeatedly computes mid-point of W_{min} and W_{max} and sets current window size to it. According to feedback

in the form of packet losses, the W_{min} or W_{max} are changed. This is continued till difference between W_{min} and W_{max} is becomes less than a defined threshold S_{min}.

Additive Increase: If the difference between the midpoint and minimum size is too large, a sudden change in window size may cause stress on the network. To avoid this, if the difference between W_{min} and midpoint is greater than a threshold value S_{max}, the window size is increased by S_{max} only.

Slow Start: When the window size crosses the W_{max} value and is less than W_{max} + S_{max}, slow start algorithm is used. Here, the window size will increase in multiples of S_{min}.
The main benefit of BIC is its concave response function. After a loss event, window size decreases to W_{max} (1-**β**), where **β** is the multiplicative decreas For a very large window, the sending rate of BIC is the same as the sending rate of AIMD with increase parameter S_{max}.

Reducing **β** increases the sending rate. Reducing **β** also improves utilization. However, it hurts convergence since larger window flows give up their bandwidth slowly. **β** has a much less impact on the sending rate than S_{max}. So it is easier to fix **β** and then adjust S_{max} and S_{min}. In the paper, **β** was chosen as 0.125 but in the Linux kernel code, it was chosen as 0.2.

According to the analysis in the paper, around small window sizes, BIC shows the worst RTT unfairness. Overall, its RTT unfairness is much better than HSTCP (High Speed) and STCP (Scalable). HSTCP and STCP tend to starve long RTT flows under high bandwidth environments.

Limitation:
One possible limitation of BIC is that as the loss rate reduces below 1e-8, its sending rate does not grow quickly. This is because of the lower slope of its response function.

4. Simulation Setup

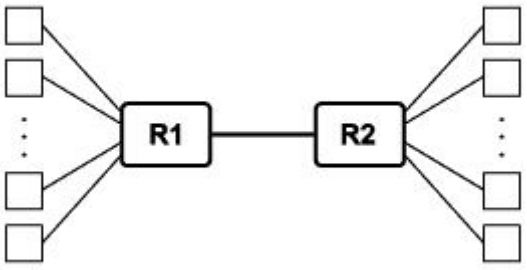


Fig. Dumbell Topology

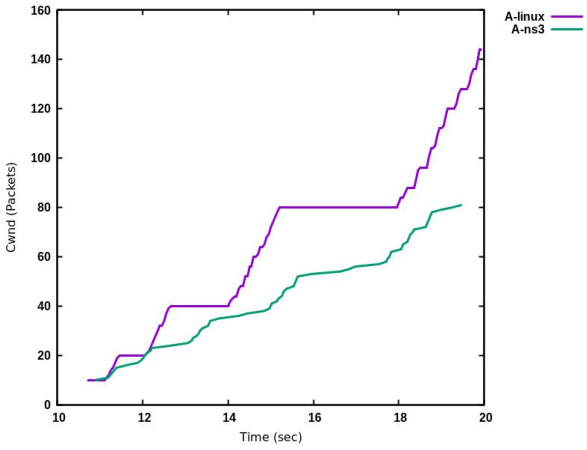
A dumbbell topology with five senders and five receivers is set up.
The routers permanently use the ns-3 stack. The simulation is run using first ns-3 and then Linux stacks on the end hosts. The bandwidth is selected such that congestion occurs at the bottleneck, ie, the two routers.
BulkSend and PacketSink applications are installed on the end nodes. Bulk send application sends as much traffic as possible, trying to fill the bandwidth and essentially keeping a constant flow of data. The PacketSink application receives and consumes traffic generated to a particular IP address and port.
Dynamic routing is done while using ns-3 stack on the hosts, and static routing is done with Linux stack.

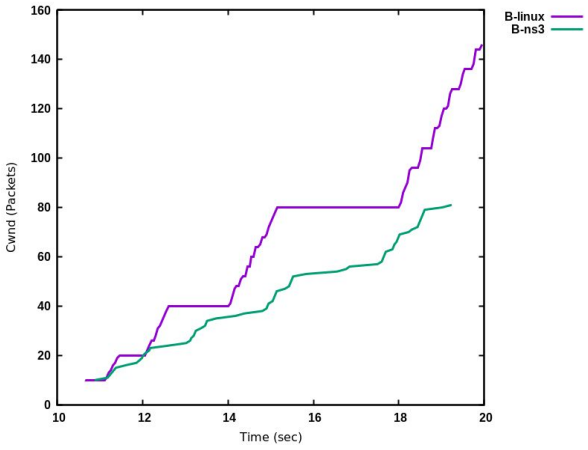
5. Results

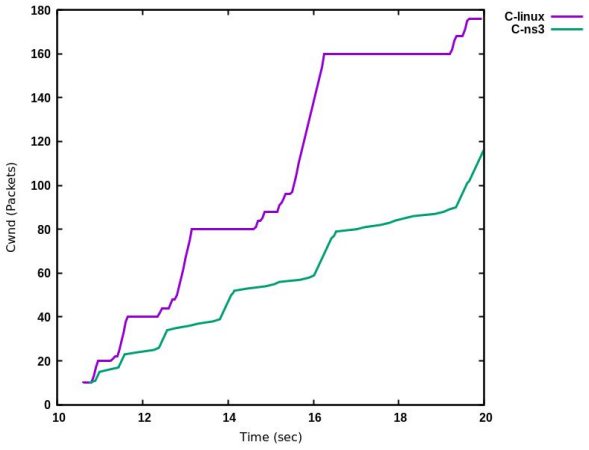
Graphs with each change:

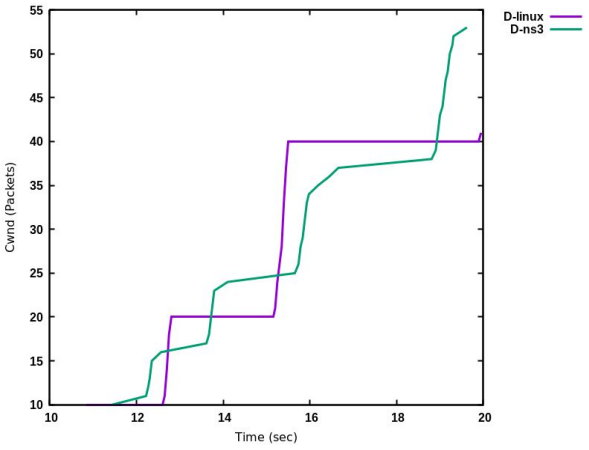
1.

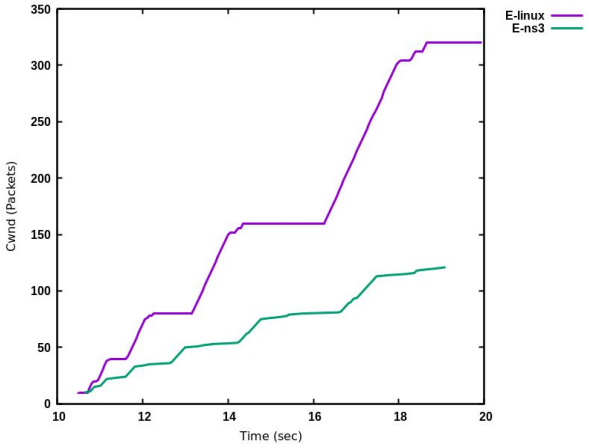
Start time	10
Stop time	20
Queue disc type	Fifo
Router-to-router bandwidth	150Mbps
Host-to-router bandwidth	150Mbps
Router-to-router Delay	0.00075ms
Host-to-router Delay	0.00025ms, 0.0001ms, 0.00005ms, 0.000025ms, 0.000005ms





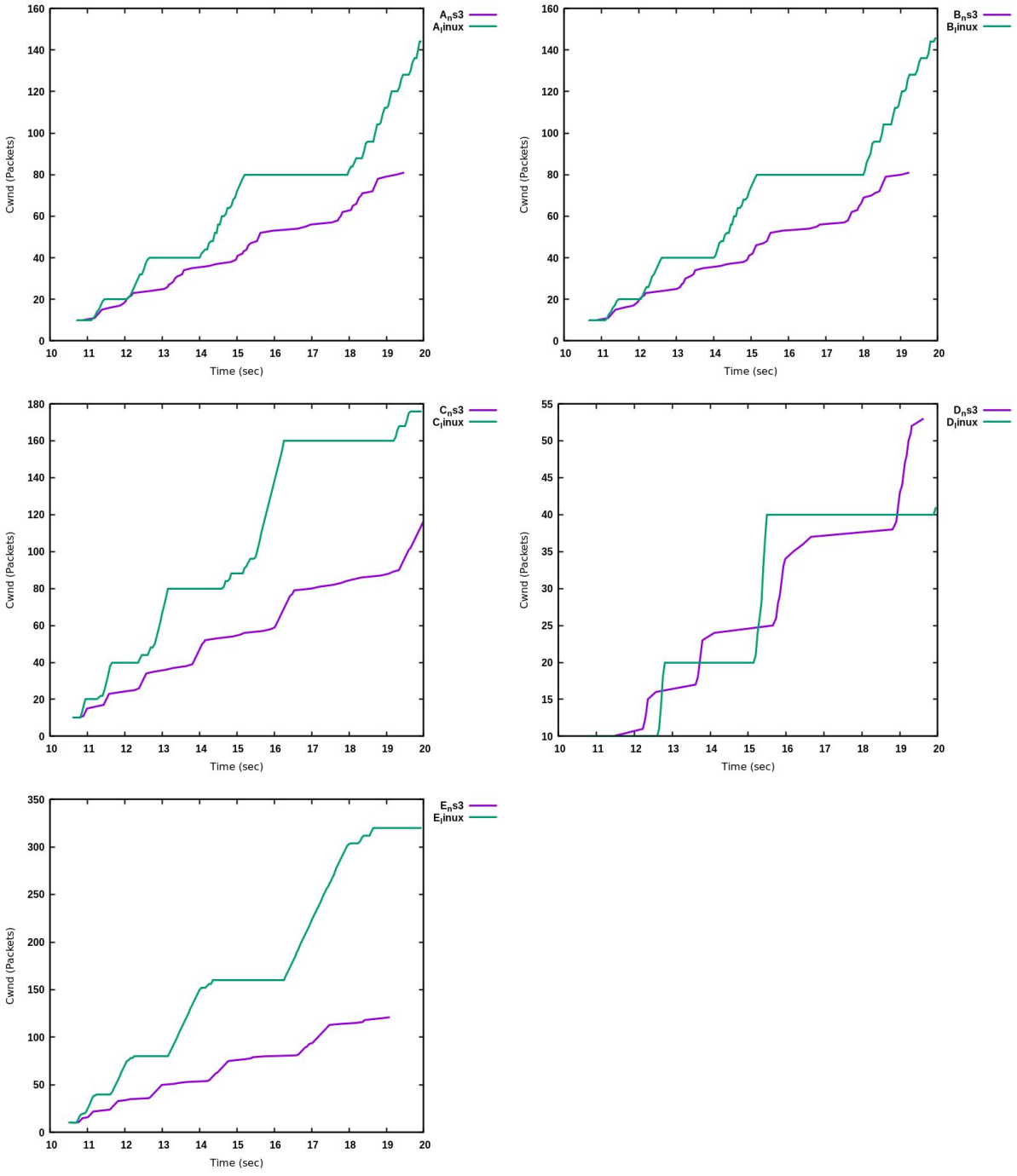






2.

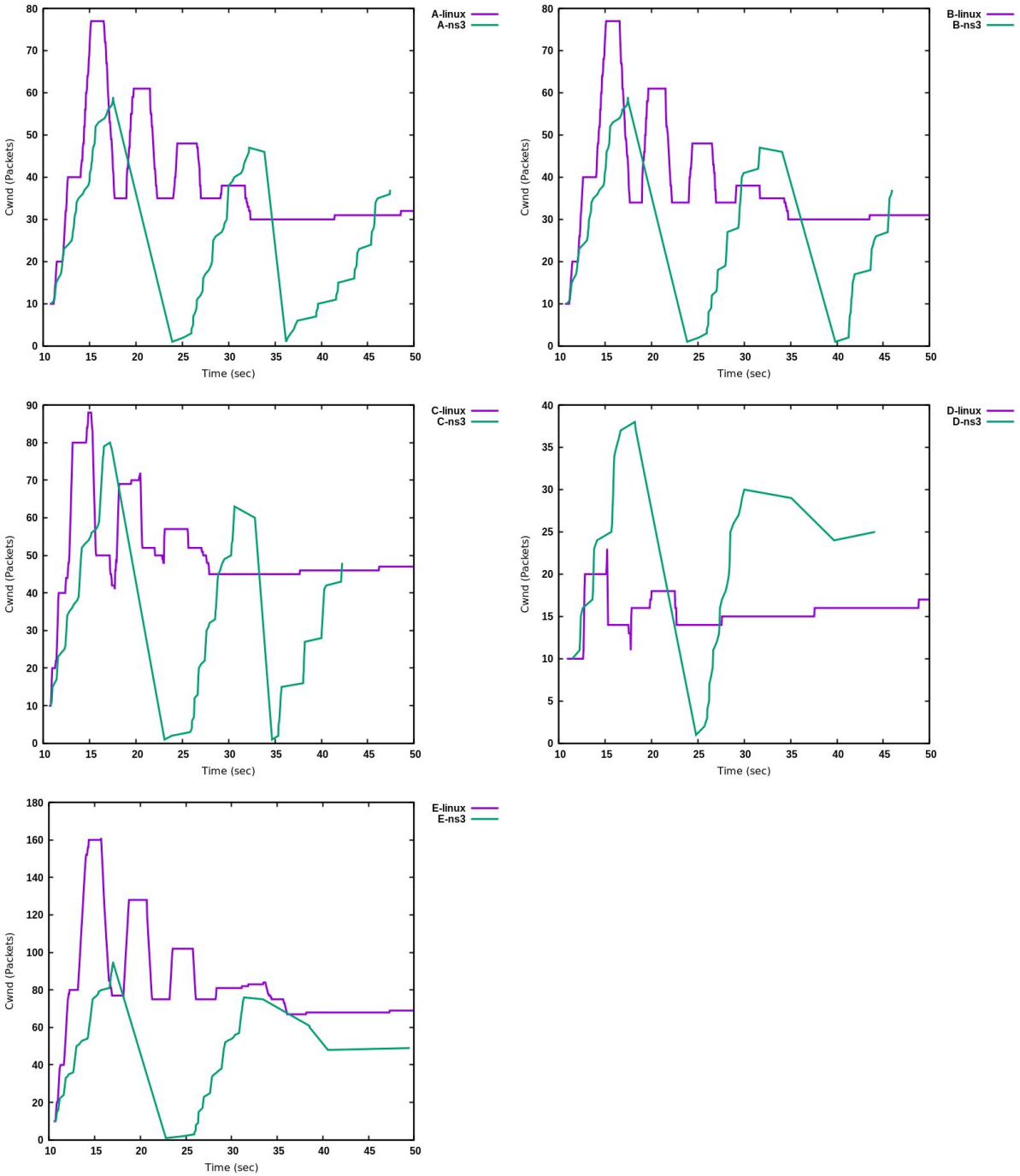
Start time	10
Stop time	20
Queue disc type	PfifoFast
Router-to-router bandwidth	150Mbps
Host-to-router bandwidth	150Mbps
Router-to-router Delay	0.00075ms
Host-to-router Delay	0.00025ms, 0.0001ms, 0.00005ms, 0.000025ms, 0.000005ms



3.

Start time	10
Stop time	50
Smooth_part in tcp-bic.cc	20
Queue disc type	PfifoFast
Router-to-router bandwidth	1Mbps
Host-to-router bandwidth	10Mbps

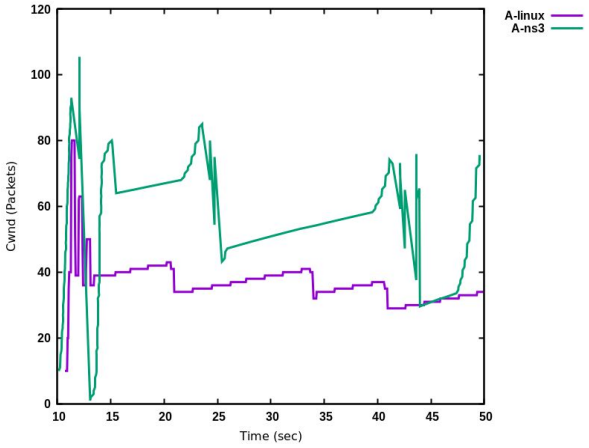
Router-to-router Delay	50ms
Host-to-router Delay	5ms

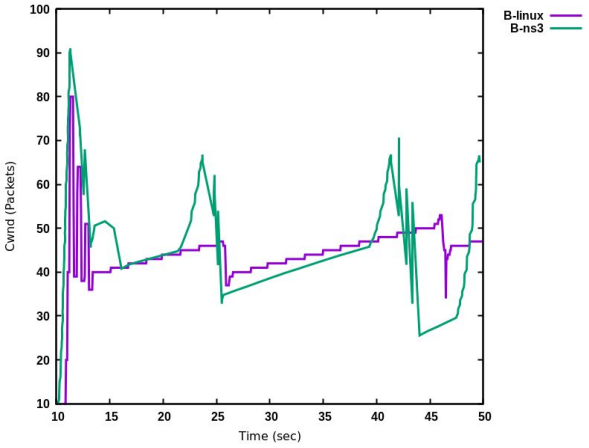


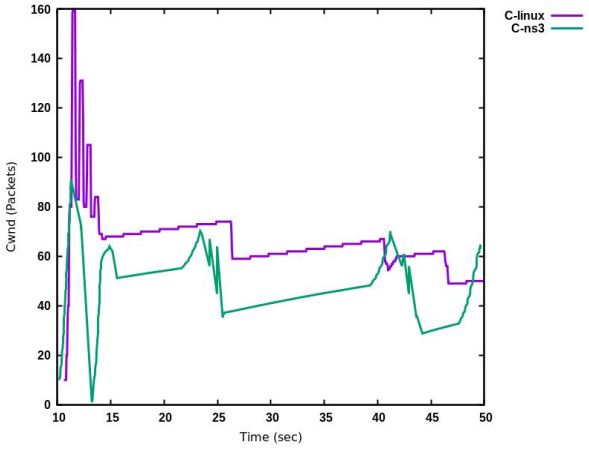
4.

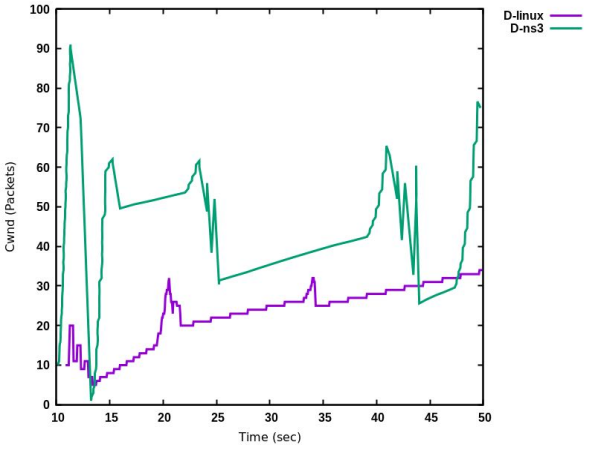
Start time	10
Stop time	50

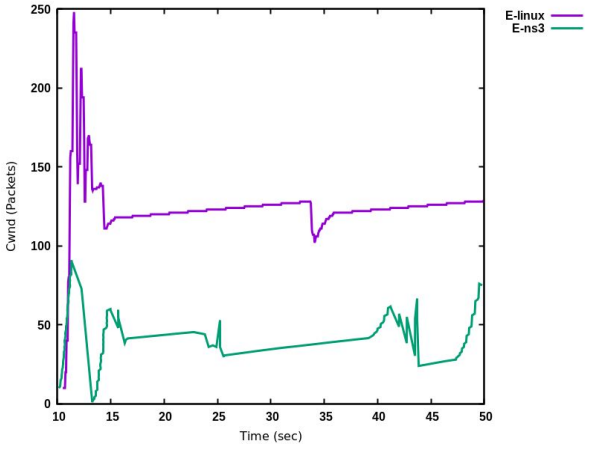
Queue disc type	PfifoFast
Router-to-router bandwidth	10Mbps
Host-to-router bandwidth	100Mbps
Router-to-router Delay	50ms
Host-to-router Delay	5ms





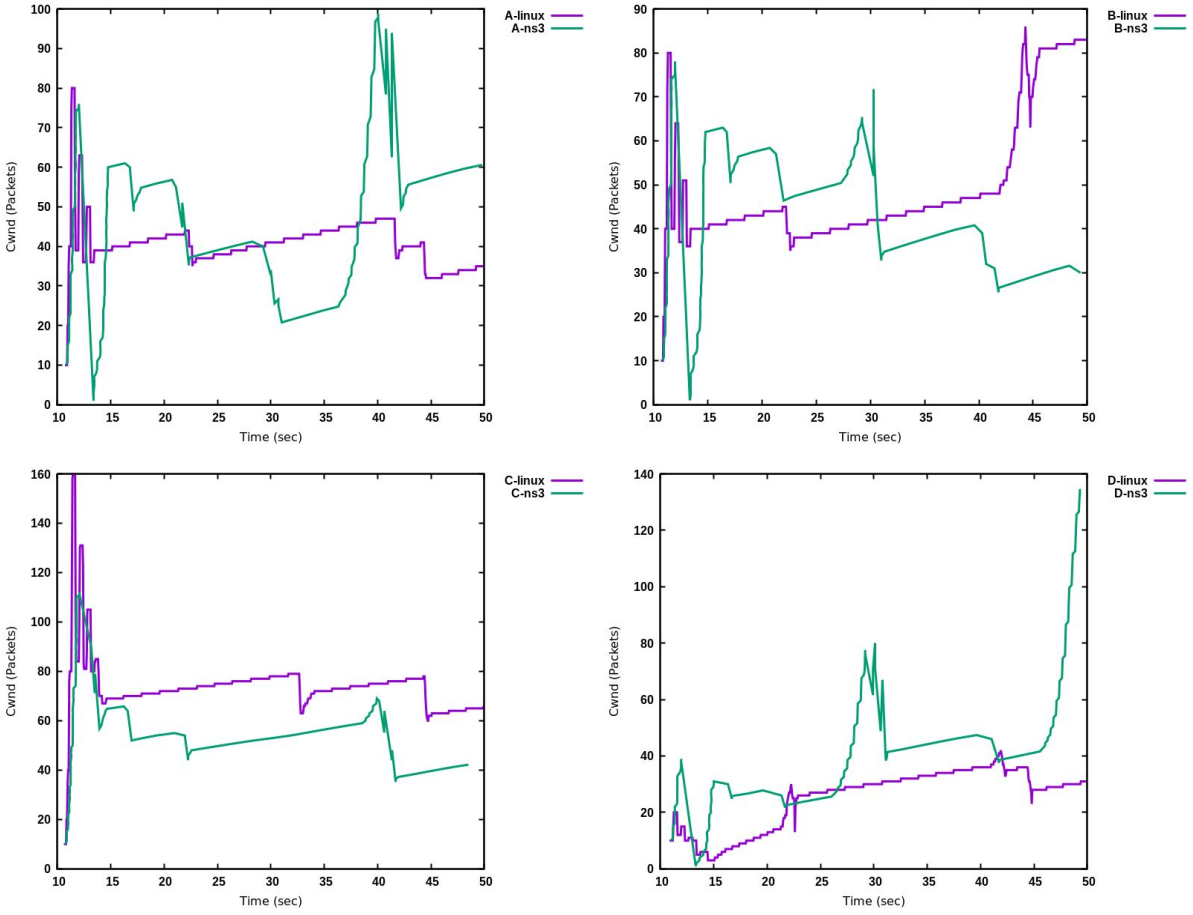


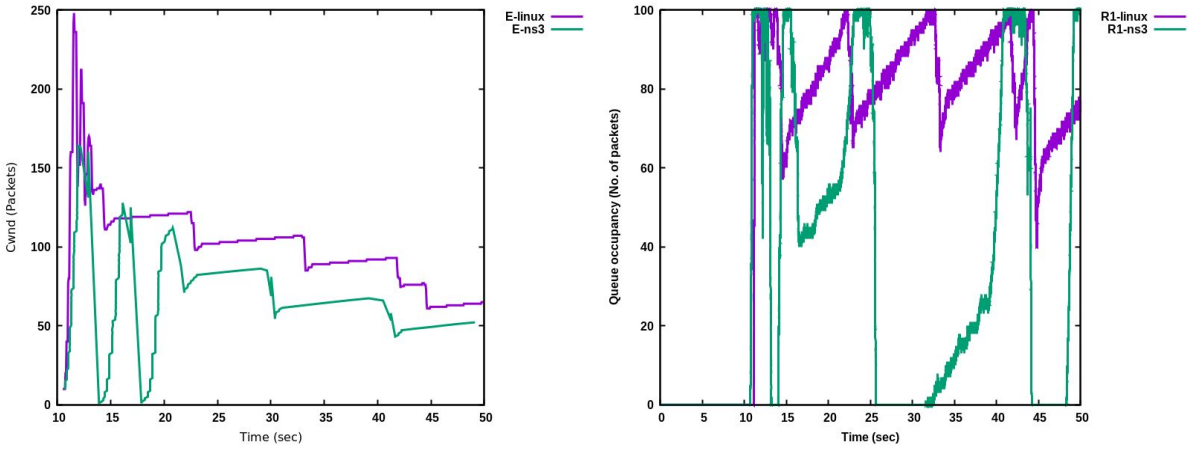




5.

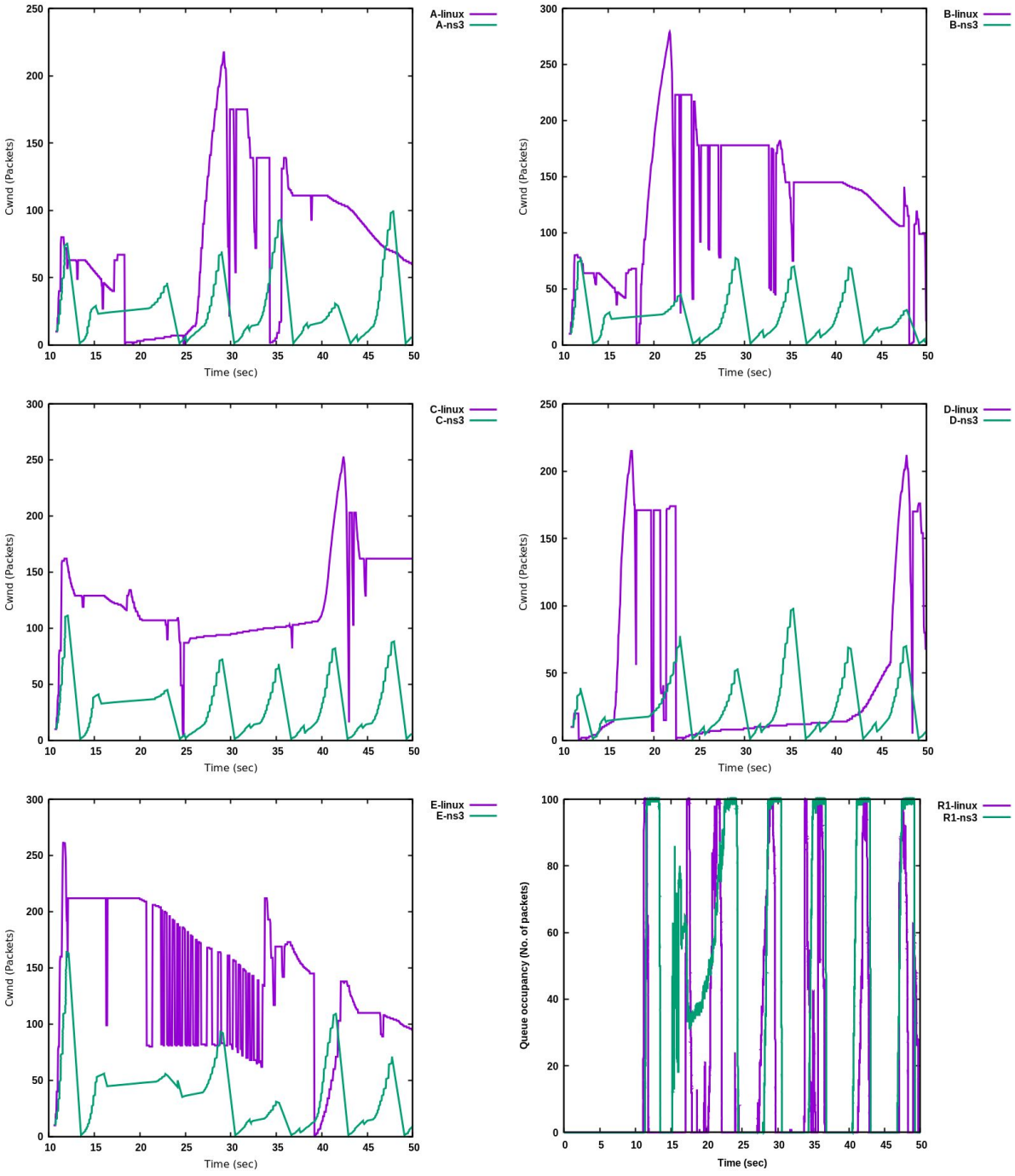
Start time	10
Stop time	50
RACK, DSACK, FACK in Linux	Disabled
Queue disc type	PfifoFast
Router-to-router bandwidth	10Mbps
Host-to-router bandwidth	100Mbps
Router-to-router Delay	50ms
Host-to-router Delay	5ms





6.

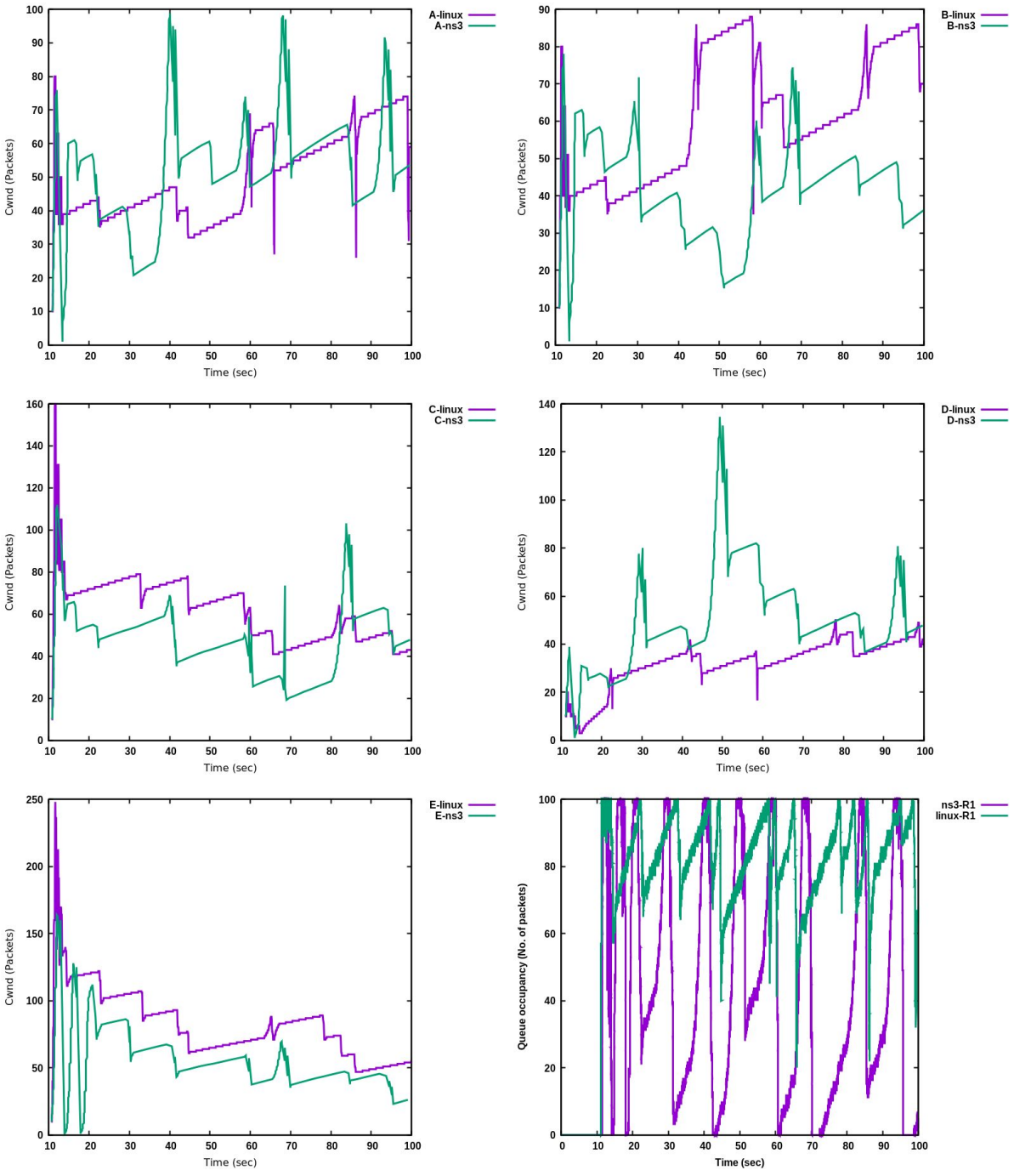
Start time	10
Stop time	50
RACK, DSACK, FACK in Linux	Disabled
SACK in ns-3 and Linux	Disabled
Queue disc type	PfifoFast
Router-to-router bandwidth	10Mbps
Host-to-router bandwidth	100Mbps
Router-to-router Delay	50ms
Host-to-router Delay	5ms



7.

Start time	10
Stop time	100
RACK, DSACK, FACK in Linux	Disabled
Queue disc type	PfifoFast
Router-to-router bandwidth	10Mbps

Host-to-router bandwidth	100Mbps
Router-to-router Delay	50ms
Host-to-router Delay	5ms



Changes made in dce-gfc-dumbbell.cc:

- Changed router to router bandwidth to 10 Mbps
- Changed node to router bandwidth to 100 Mbps
- Router to router delay changed to 50 ms

- Changed node to router delay to 5 ms
- Changed the queue to PfifoFastQueue
- Changed queue size to 100.
- Disabled FACK, DSACK and RACK in Linux, as these are not implemented in ns-3
- Disabled SACK in Linux and ns-3

Changes made in ns-3-dev/src/internet/model/tcp-bic.cc

- Changed smooth_part parameter to 20

6. Takeaways and future work

- Understand the algorithm and working of TCP BIC, which reduces the time required to recover the reduced window size to logarithmic time.
- Comparison of Linux kernel code and ns-3 implementations of TCP BIC.
- Worked with new open source software.

The congestion window and queue size graphs obtained thus far for Linux and ns-3 stacks are not exactly the same. This points to possible discrepancies between the ns-3 and Linux implementations of BIC TCP. Thus, further work is required to bring the graphs to coincide exactly. This involves iterations by trial and error.