

1 What is NS-3 DCE?

Direct Code Execution (DCE) is a framework for ns-3 that provides facilities to execute, within ns-3, existing implementations of userspace and kernelspace network protocols or applications without source code changes. For example, instead of using ns-3's implementation of a ping-like application, you can use the real ping application. You can also use the Linux networking stack in simulations.

for more info please visit :

<https://www.nsnam.org/about/projects/direct-code-execution/>

<https://www.nsnam.org/docs/dce/manual/html/getting-started.html>

<https://www.nsnam.org/docs/dce/manual/ns-3-dce-manual.pdf>

2 Why a Docker Image of NS-3 DCE instead of normal installation?

- 1 NS-3 DCE direct installation is supported till Ubuntu 14.04(as per the above documentation), and is extended to 16.04 as well. Still this installation is very platform specific and a lot of dependencies needs to be resolved while a standard installation.
- 2 It has a well known dependency issue (Glib) which makes it incompatible with Ubuntu 18.04 LTS.
- 3 Since Docker container wraps up all the necessary dependencies together, we can easily create and use the DCE image independent of underlying OS.

We started off with installing DCE on UBUNTU 18.04 and then on 16.04, and due to above reasons, we switched to a docker image. Later we found this approach very helpful as we can create as many containers as we want (which is nothing but different copies of ns-3 DCE) and we can experiment different techniques in different containers simultaneously on one System. Also, since we are also working with ns3-dev version, it is quite flexible to build dce on different containers independently.

3 Installing and Setting Up DCE

After installing docker follow the steps to setup DCE:

1. Pull the docker image to install NS-3 DCE.

```
# docker pull thehajime/ns-3-dce
```

2. Run docker to create a container.

```
# sudo docker run -i -t thehajime/ns-3-dce /bin/bash
```

This will take you to a particular container where you can find the ns-3 dce.

3. Install vim in docker container.

```
# sudo apt install vim
```

```

ns3dce@133a04e11c7b: ~/dce-linux-dev/source
dumbbell-topology-cubic-variant.cc dumbbell-topology-ns3-receiver.cc
ns3dce@133a04e11c7b:~/apoorva$ cp tcp_testing_and_alignment/Examples/dumbbell-topology-ns3-receiver.cc ./
.bakerc .bash_history .bash_logout .bashrc .ccache/ .profile .rnd apoorva/ dce-linux-dev/
ns3dce@133a04e11c7b:~/apoorva$ cp tcp_testing_and_alignment/Examples/dumbbell-topology-ns3-receiver.cc ./dce-linux-dev/
.bakerc .bash_history .bash_logout .bashrc .ccache/ .profile .rnd apoorva/ dce-linux-dev/
ns3dce@133a04e11c7b:~/apoorva$ cp tcp_testing_and_alignment/Examples/dumbbell-topology-ns3-receiver.cc ./dce-linux-dev/source/n
Config bake/ bakeSetEnv.sh bakefile.xml build/ source/
ns3dce@133a04e11c7b:~/apoorva$ cp tcp_testing_and_alignment/Examples/dumbbell-topology-ns3-receiver.cc ./dce-linux-dev/source/n
net-next-nuse-4.4.0/ netanin/ ns-3-dce/ ns-3-dev/
ns3dce@133a04e11c7b:~/apoorva$ cp tcp_testing_and_alignment/Examples/dumbbell-topology-ns3-receiver.cc ./dce-linux-dev/source/ns-3-dce/
.circlect/ elf-cache/ test/
.git/ example/ test.py
.github/ exitprocs testpy-output/
.gitignore files-0/ testpy-suppl
.hgtags files-1/ utils/
.lock-waf_linux2_build gpl-2.0.txt utils.py
.waf-1.8.19-b1fc8f7baef51bd2db4c2971909a568d/ helper/ utils.pyc
AUTHORS iperf-ns3-0-0.pcap waf
README.md model/ waf-tools/
RELEASE_NOTES nyscripts/ wscript
bindings/ netlink/ wutlis.py
build/ ns3waf/ wutlis.pyc
doc/ objdir/
ns3dce@133a04e11c7b:~/apoorva$ cp tcp_testing_and_alignment/Examples/dumbbell-topology-ns3-receiver.cc ./dce-linux-dev/source/ns-3-dce/ex
example/ exitprocs
ns3dce@133a04e11c7b:~/apoorva$ cp tcp_testing_and_alignment/Examples/dumbbell-topology-ns3-receiver.cc ./dce-linux-dev/source/ns-3-dce/exampl
e/
ns3dce@133a04e11c7b:~/apoorva$ cd ../
ns3dce@133a04e11c7b:~$ cd
.bakerc .bash_history .bash_logout .bashrc .ccache/ .profile .rnd apoorva/ dce-linux-dev/
ns3dce@133a04e11c7b:~$ cd dce-linux-dev/
Config bake/ bakeSetEnv.sh bakefile.xml build/ source/
ns3dce@133a04e11c7b:~$ cd dce-linux-dev/
Config bake/ bakeSetEnv.sh bakefile.xml build/ source/
ns3dce@133a04e11c7b:~$ cd dce-linux-dev/source/n
net-next-nuse-4.4.0/ netanin/ ns-3-dce/ ns-3-dev/
ns3dce@133a04e11c7b:~$ cd dce-linux-dev/source/ns-3-dce/

```

4. Go to source

`cd source`

Ns-3-dce // Direct code execution section

Ns-3-dev // Development section

5. go to ns-3-dce and run following command to check weather install correctly or not(by running iperf)

`cd ns-3-dce`

`./waf --run dce-iperf`

If build successfully : You have correctly installed DCE!

```

ns3dce@8823856d0edc:~/dce-linux-dev$ ./waf --run dce-iperf
bash: ./waf: No such file or directory
ns3dce@8823856d0edc:~/dce-linux-dev$ pwd
/home/ns3dce/dce-linux-dev
ns3dce@8823856d0edc:~/dce-linux-dev$ cd source/ns-3-dce/
ns3dce@8823856d0edc:~/dce-linux-dev/source/ns-3-dce$ ./waf --run dce-iperf
waf: Entering directory '/home/ns3dce/dce-linux-dev/source/ns-3-dce/build'
[ 12/413] Creating build/lib/pkgconfig/libns3-dev-netlink-debug.pc
[ 118/413] Creating build/lib/pkgconfig/libns3-dev-dce-debug.pc
[ 120/413] Creating build/myscripts/ns-3-dce-unip/lib/pkgconfig/libns3-dev-dce-unip-debug.pc
[ 126/413] Creating build/myscripts/ns-3-dce-quagga/lib/pkgconfig/libns3-dev-dce-quagga-debug.pc
waf: Leaving directory '/home/ns3dce/dce-linux-dev/source/ns-3-dce/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1.384s)
ns3dce@8823856d0edc:~/dce-linux-dev/source/ns-3-dce$ cat files-1/var/log/*/stdout
.....
Server listening on TCP port 5001
TCP window size: 128 KByte (default)
.....
[ 4] local 10.1.1.2 port 5001 connected with 10.1.1.1 port 49153
[ ID] Interval Transfer Bandwidth
[ 4] 0.0-11.2 sec 5.62 MBytes 4.23 Mbits/sec
ns3dce@8823856d0edc:~/dce-linux-dev/source/ns-3-dce$

```

6. Copy dumbbell topology in ns-3-dce

```
# sudo docker cp dumbbell-topology-ns3-receiver.cc your_docker_name:/home/ns3dce/dce-linux-dev/source/ns-3-dce/example
```

The file to be copied must be in home directory on local machine. Run the above command on a different terminal outside the container.

- Go to your wscript and make entry of this dumbbell-topology-ns3-receiver file in example.

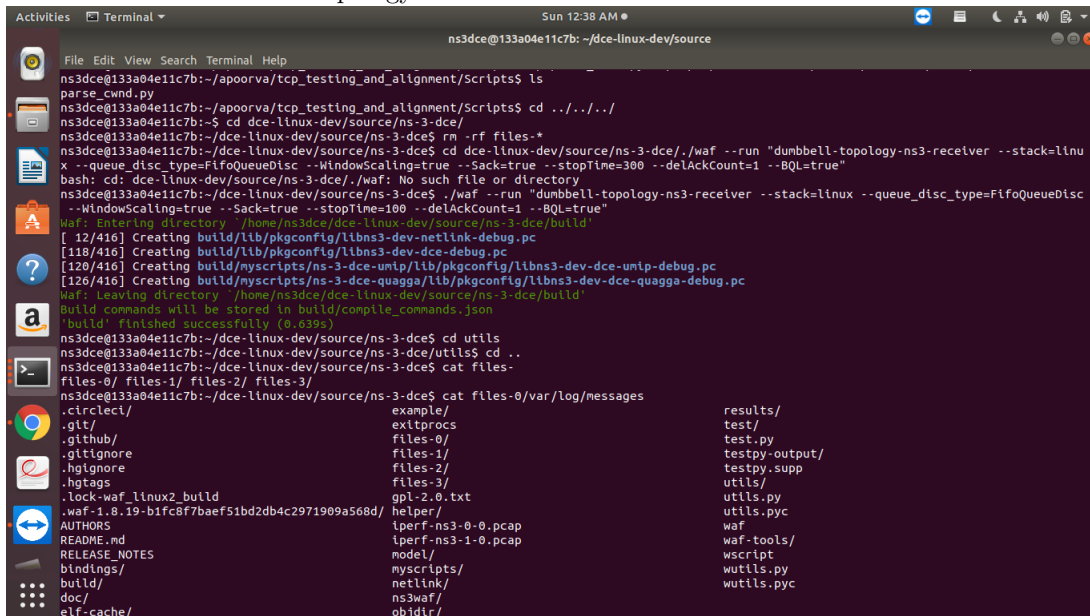
```
cd /home/ns3dce/dce-linux-dev/source/ns-3-dce
```

wscript will be present at above location.

- Run the example file on linux stack.

```
./waf --run "dumbbell-topology-ns3-receiver --stack=linux -queue_disc_type=FifoQueueDisc --WindowScaling=true --Sack=true --stopTime=300 --delAckCount=1 --BQL=true"
```

This will run the dumbbell topology on linux stack.



```
ns3dce@133a04e11c7b: ~/dce-linux-dev/source
ns3dce@133a04e11c7b:~/apoorva/tcp_testing_and_alignment/Scripts$ ls
parse_cwnd.py
ns3dce@133a04e11c7b:~/apoorva/tcp_testing_and_alignment/Scripts$ cd ../../../../
ns3dce@133a04e11c7b:~/dce-linux-dev/source/ns-3-dce$ rm -rf files.*
ns3dce@133a04e11c7b:~/dce-linux-dev/source/ns-3-dce$ cd dce-linux-dev/source/ns-3-dce/./waf --run "dumbbell-topology-ns3-receiver --stack=linux
x -queue_disc_type=FifoQueueDisc --WindowScaling=true --Sack=true --stopTime=300 --delAckCount=1 --BQL=true"
bash: cd: dce-linux-dev/source/ns-3-dce/./waf: No such file or directory
ns3dce@133a04e11c7b:~/dce-linux-dev/source/ns-3-dce$ ./waf --run "dumbbell-topology-ns3-receiver --stack=linux --queue_disc_type=FifoQueueDisc
--WindowScaling=true --Sack=true --stopTime=300 --delAckCount=1 --BQL=true"
waf: Entering directory '/home/ns3dce/dce-linux-dev/source/ns-3-dce/build'
[ 12/416] Creating build/lib/pkgconfig/libns3-dev-netlink-debug.pc
[ 18/416] Creating build/lib/pkgconfig/libns3-dev-dce-debug.pc
[ 20/416] Creating build/myscripts/ns-3-dce-umtp/lib/pkgconfig/libns3-dev-dce-umtp-debug.pc
[ 26/416] Creating build/myscripts/ns-3-dce-quagga/lib/pkgconfig/libns3-dev-dce-quagga-debug.pc
waf: Leaving directory '/home/ns3dce/dce-linux-dev/source/ns-3-dce/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.639s)
ns3dce@133a04e11c7b:~/dce-linux-dev/source/ns-3-dce$ cd utils
ns3dce@133a04e11c7b:~/dce-linux-dev/source/ns-3-dce/utils$ cd ..
ns3dce@133a04e11c7b:~/dce-linux-dev/source/ns-3-dce$ cat files-
files-0/ files-1/ files-2/ files-3/
ns3dce@133a04e11c7b:~/dce-linux-dev/source/ns-3-dce$ cat files-0/var/log/messages
.circleci/
.git/
.github/
.gitignore
.hgignore
.hgtags
.lock-waf_linux2_build
.waf-1.8.19-b1fc8f7baef51bd2db4c2971909a568d/
AUTHORS
README.md
RELEASE_NOTES
bindings/
build/
doc/
elf-cache/
example/
exitprocs
files-0/
files-1/
files-2/
files-3/
gpl-2.0.txt
helper/
lperf-ns3-0-0.pcap
lperf-ns3-1-0.pcap
model/
myscripts/
netlink/
ns3waf/
objdir/
results/
test/
test.py
testpy-output/
testpy-suppl
utils/
utils.py
utils.pyc
waf
waf-tools/
wscript
utils.py
utils.pyc
```

- Copy the parse_cwnd.py file in ns-3-dce/utils.

```
# sudo docker cp parse_cwnd.py your_docker_name:/home/ns3dce/dce-linux-dev/source/ns-3-dce/utils
```

- Run parse_cwnd inside ns-3-dce/utils.

```
python parse_cwnd.py 2 2
```

Running this file will collect the traces generated by running the topology example on linux stack. This will create a folder 'cwnd_data' inside ns-3-dce/result/dumbbell-topology where you will find A-linux.plotme file.

11. Make one folder "overlapped" inside ns-3-dce/result/dumbbell-topology.

```
# sudo mkdir overlapped
```

12. copy the A-linux.plotme file from cwnd_data to overlapped folder.

```
cp cwnd_data/A-linux.plotme overlapped/
```

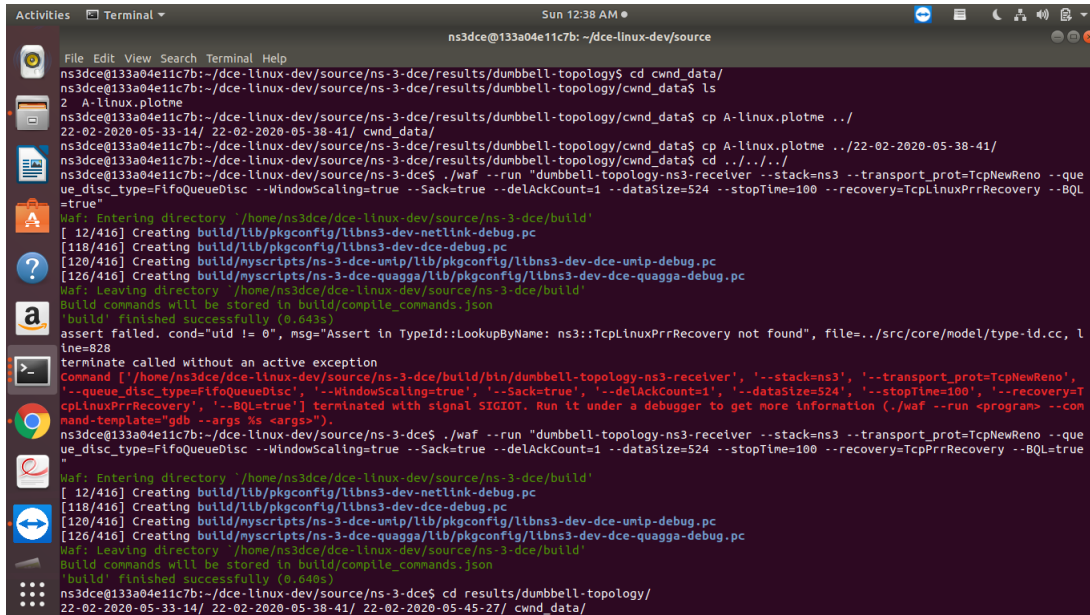
This should be done from docker container itself.

13. Run the example topology on ns-3 stack.

```
./waf --run "dumbbell-topology ns3-receiver --stack=ns3 --queue_disc_type=FifoQueueDisc --WindowScaling=true  
--Sack=true --stopTime=300 --delAckCount=1 --BQL=true --transport_prot=TcpNewReno"
```

Run this from source/ns-3-dce itself.

This will create a new timestamp folder under ns-3-dce/result/dumbbell-topology/.



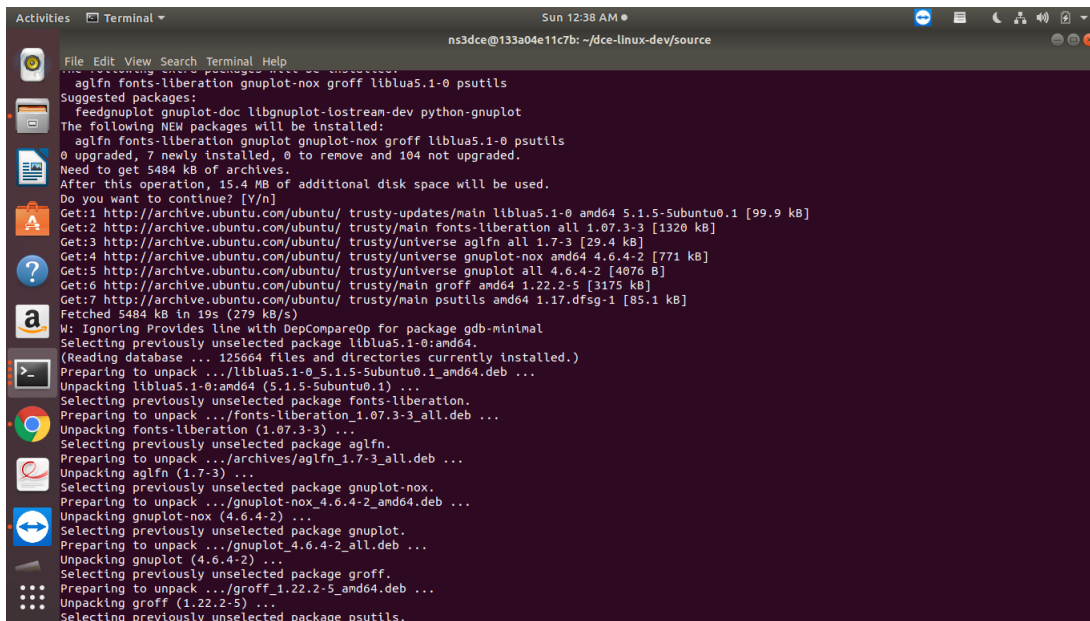
```
Activities Terminal
Sun 12:38 AM
ns3dce@133a04e11c7b: ~/dce-linux-dev/source
ns3dce@133a04e11c7b:~/dce-linux-dev/source/ns-3-dce/results/dumbbell-topology$ cd cwnd_data/
ns3dce@133a04e11c7b:~/dce-linux-dev/source/ns-3-dce/results/dumbbell-topology/cwnd_data$ ls
2 A-linux.plotme
ns3dce@133a04e11c7b:~/dce-linux-dev/source/ns-3-dce/results/dumbbell-topology/cwnd_data$ cp A-linux.plotme ../
22-02-2020-05-33-14/ 22-02-2020-05-38-41/ cwnd_data/
ns3dce@133a04e11c7b:~/dce-linux-dev/source/ns-3-dce/results/dumbbell-topology/cwnd_data$ cd ../../
ns3dce@133a04e11c7b:~/dce-linux-dev/source/ns-3-dce$ ./waf --run "dumbbell-topology ns3-receiver --stack=ns3 --transport_prot=TcpNewReno --que
ue_disc_type=FifoQueueDisc --WindowScaling=true --Sack=true --delAckCount=1 --dataSize=524 --stopTime=100 --recovery=TcpLinuxPrrRecovery --BQL
=true"
Waf: Entering directory '/home/ns3dce/dce-linux-dev/source/ns-3-dce/build'
[ 12/416] Creating build/lib/pkgconfig/libns3-dev-netlink-debug.pc
[118/416] Creating build/lib/pkgconfig/libns3-dev-dce-debug.pc
[120/416] Creating build/nyscripts/ns-3-dce-unimp/lib/pkgconfig/libns3-dev-dce-unimp-debug.pc
[126/416] Creating build/nyscripts/ns-3-dce-quagga/lib/pkgconfig/libns3-dev-dce-quagga-debug.pc
Waf: Leaving directory '/home/ns3dce/dce-linux-dev/source/ns-3-dce/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.643s)
assert failed. cond="uid != 0", msg="Assert in TypeId::LookupByName: ns3::TcpLinuxPrrRecovery not found", file=../src/core/model/type-id.cc, l
ine=828
terminate called without an active exception
Command ['/home/ns3dce/dce-linux-dev/source/ns-3-dce/build/bin/dumbbell-topology-ns3-receiver', '--stack=ns3', '--transport_prot=TcpNewReno',
'--queue_disc_type=FifoQueueDisc', '--WindowScaling=true', '--Sack=true', '--delAckCount=1', '--dataSize=524', '--stopTime=100', '--recovery=T
cpLinuxPrrRecovery', '--BQL=true'] terminated with signal SIGIOT. Run it under a debugger to get more information (./waf --run <program> --con
mand-template=gdb --args %s <args>).
ns3dce@133a04e11c7b:~/dce-linux-dev/source/ns-3-dce$ ./waf --run "dumbbell-topology ns3-receiver --stack=ns3 --transport_prot=TcpNewReno --que
ue_disc_type=FifoQueueDisc --WindowScaling=true --Sack=true --delAckCount=1 --dataSize=524 --stopTime=100 --recovery=TcpLinuxPrrRecovery --BQL=true"
Waf: Entering directory '/home/ns3dce/dce-linux-dev/source/ns-3-dce/build'
[ 12/416] Creating build/lib/pkgconfig/libns3-dev-netlink-debug.pc
[118/416] Creating build/lib/pkgconfig/libns3-dev-dce-debug.pc
[120/416] Creating build/nyscripts/ns-3-dce-unimp/lib/pkgconfig/libns3-dev-dce-unimp-debug.pc
[126/416] Creating build/nyscripts/ns-3-dce-quagga/lib/pkgconfig/libns3-dev-dce-quagga-debug.pc
Waf: Leaving directory '/home/ns3dce/dce-linux-dev/source/ns-3-dce/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.649s)
ns3dce@133a04e11c7b:~/dce-linux-dev/source/ns-3-dce$ cd results/dumbbell-topology/
22-02-2020-05-33-14/ 22-02-2020-05-38-41/ 22-02-2020-05-45-27/ cwnd_data/
```

14. Under the latest timestamp folder, you will find a folder "cwndTraces" where A-ns3.plotme file will be generated. Copy this file in overlapped folder.

15. Overlapped will now have A-ns3.plotme and A-linux.plotme now copy the gnuplotscriptCwnd script inside overlapped and install gnuplot.

```
#sudo apt-get install gnuplot
gnuplot overlapgnuplotscriptCwnd
```

Above steps will create a CwndA.png file inside overlapped.



```
ns3dce@133a04e11c7b: ~/dce-linux-dev/source
$ sudo apt-get install aglfn fonts-liberation gnuplot-nox groff liblua5.1-0 psutils
Reading package lists... Done
Building dependency tree... Done
The following NEW packages will be installed:
  aglfn fonts-liberation gnuplot-nox groff liblua5.1-0 psutils
0 upgraded, 7 newly installed, 0 to remove and 104 not upgraded.
Need to get 5484 kB of archives.
After this operation, 15.4 MB of additional disk space will be used.
Do you want to continue? [Y/n]
Get:1 http://archive.ubuntu.com/ubuntu/ trusty-updates/main liblua5.1-0 amd64 5.1.5-Subuntu0.1 [99.9 kB]
Get:2 http://archive.ubuntu.com/ubuntu/ trusty/main fonts-liberation all 1.07.3-3 [1320 kB]
Get:3 http://archive.ubuntu.com/ubuntu/ trusty/universe aglfn all 1.7-3 [29.4 kB]
Get:4 http://archive.ubuntu.com/ubuntu/ trusty/universe gnuplot-nox amd64 4.6.4-2 [771 kB]
Get:5 http://archive.ubuntu.com/ubuntu/ trusty/universe gnuplot all 4.6.4-2 [4076 B]
Get:6 http://archive.ubuntu.com/ubuntu/ trusty/main groff amd64 1.22.2-5 [3175 kB]
Get:7 http://archive.ubuntu.com/ubuntu/ trusty/main psutils amd64 1.17.dfsg-1 [85.1 kB]
Fetched 5484 kB in 49s (279 kB/s)
W: Ignoring Provides line with DepCompareOp for package gdb-minimal
Selecting previously unselected package liblua5.1-0:amd64.
(Reading database ... 125664 files and directories currently installed.)
Preparing to unpack .../liblua5.1-0_5.1.5-Subuntu0.1_amd64.deb ...
Unpacking liblua5.1-0:amd64 (5.1.5-Subuntu0.1) ...
Selecting previously unselected package fonts-liberation.
Preparing to unpack .../fonts-liberation_1.07.3-3_all.deb ...
Unpacking fonts-liberation (1.07.3-3) ...
Selecting previously unselected package aglfn.
Preparing to unpack .../archives/aglfn_1.7-3_all.deb ...
Unpacking aglfn (1.7-3) ...
Selecting previously unselected package gnuplot-nox.
Preparing to unpack .../gnuplot-nox_4.6.4-2_amd64.deb ...
Unpacking gnuplot-nox (4.6.4-2) ...
Selecting previously unselected package gnuplot.
Preparing to unpack .../gnuplot_4.6.4-2_all.deb ...
Unpacking gnuplot (4.6.4-2) ...
Selecting previously unselected package groff.
Preparing to unpack .../groff_1.22.2-5_amd64.deb ...
Unpacking groff (1.22.2-5) ...
Selecting previously unselected package psutils.
```

16. copy CwndA.png to Home.

```
# sudo docker cp dockername:/home/ns3dce/dce-linux-dev/source/ns-3-dce/results/dumbbell-topology/overlapped/CwndA.png .
```

Here we have got our result file inside local machine's home directory.