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
   ```

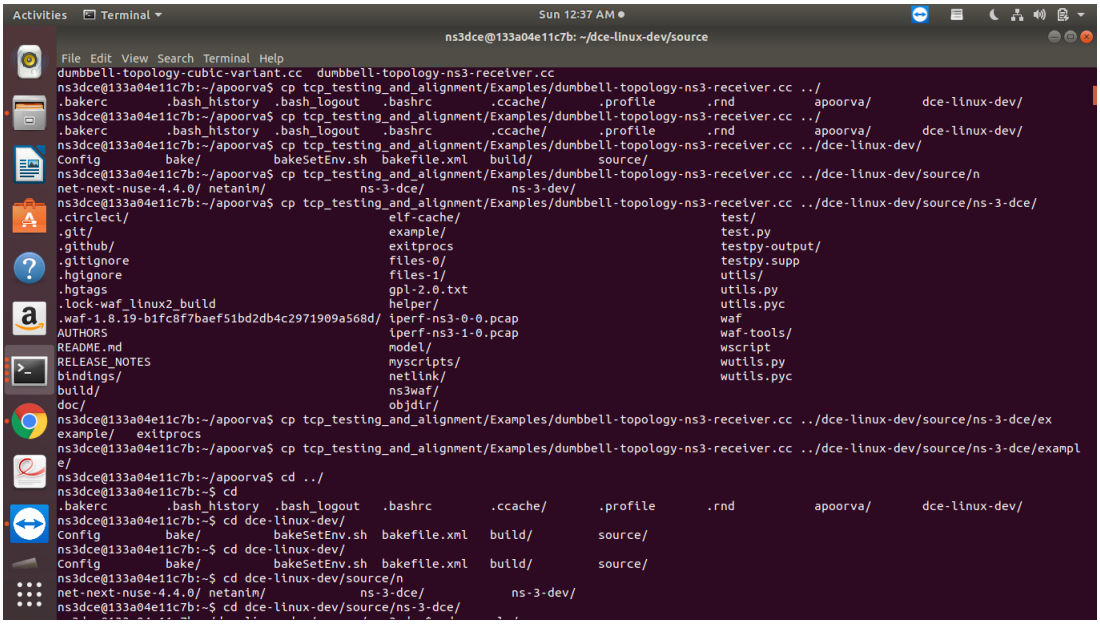
2. Run docker to create a container.

   ```
   # 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
   ```



4. Go to source

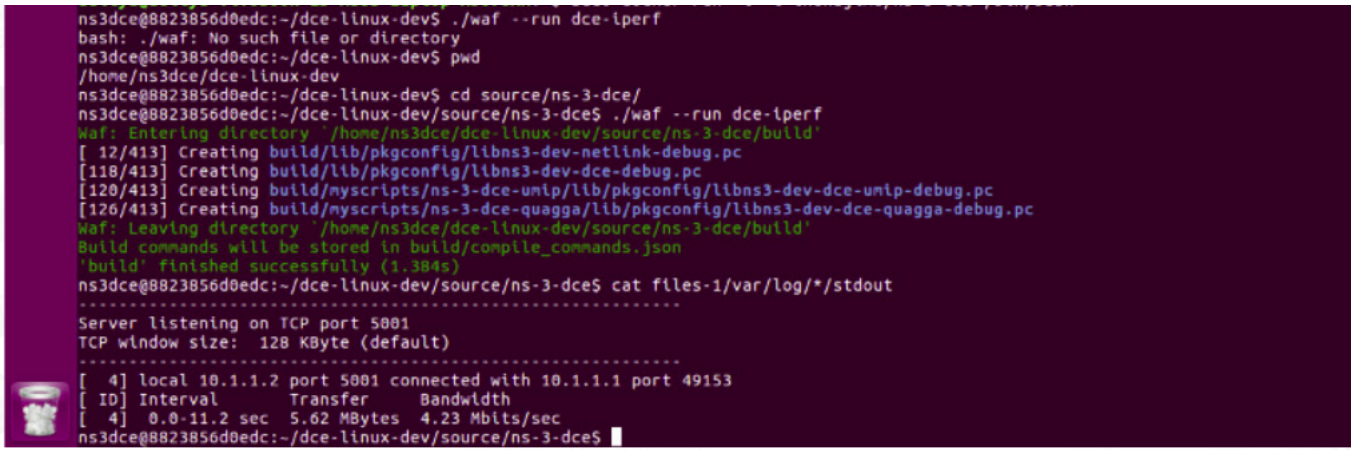
   cd source

   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!



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 ion local machine. Run the above command on a different terminal outside the container.

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

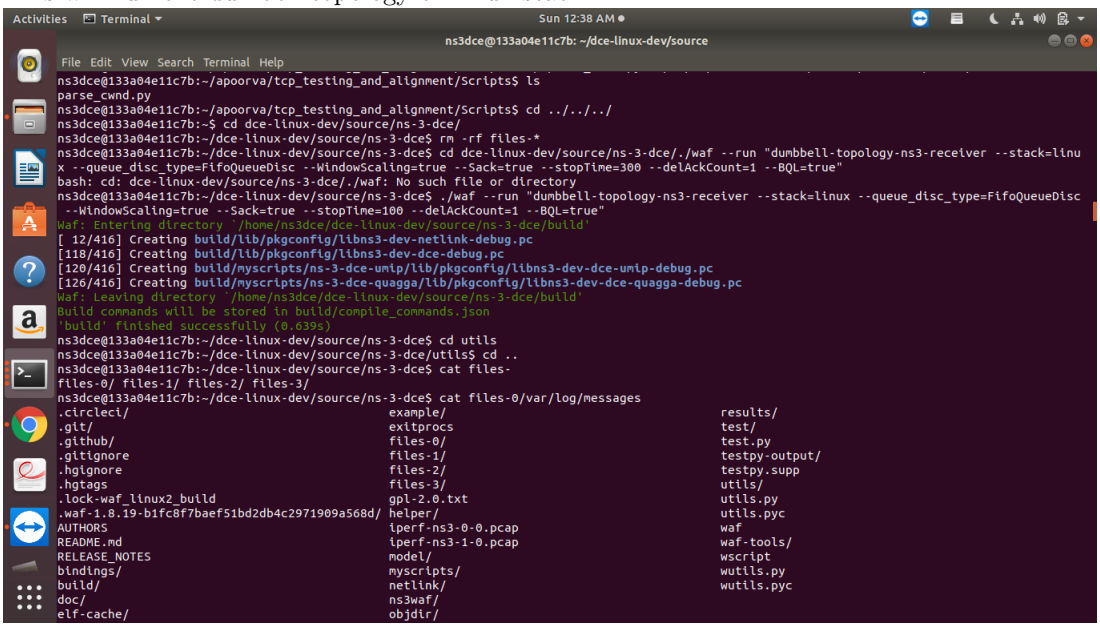
   cd /home/ns3dce/dce-linux-dev/source/ns-3-dce

   wscript will be present at above location.

8. 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 dumbell topology on linux stack.



9. Copy the parse_dcwnd.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

10. 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/dumbell-topology.

    # sudo mkdir overlapped

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

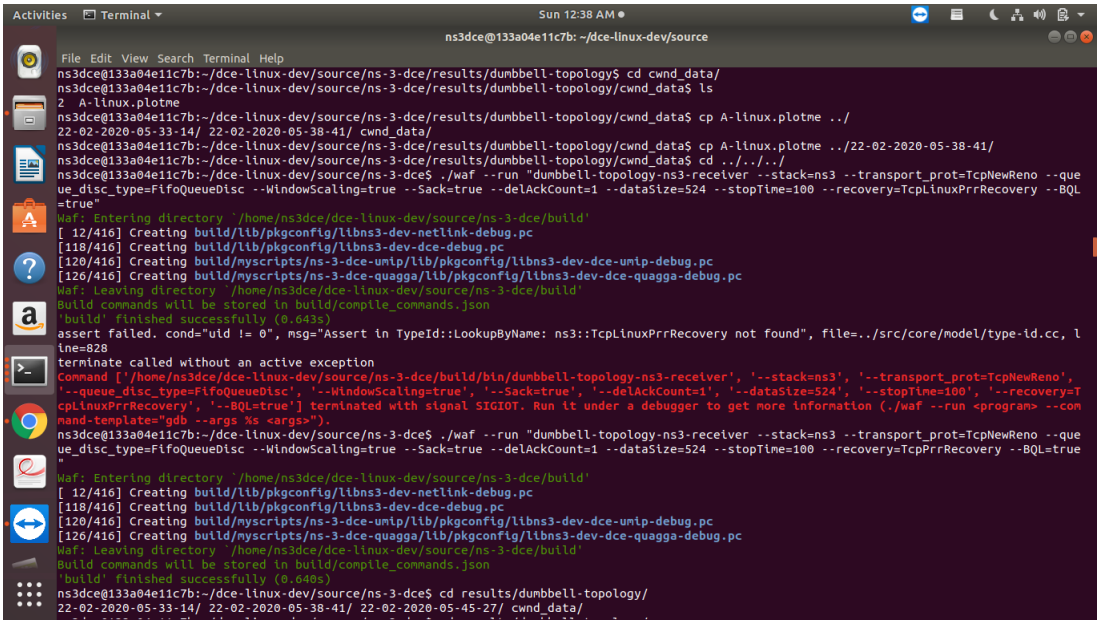
    cp cwnd_data/A-linux.plotme overlapped/

    This should be done from docker container itself.

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

    ./waf –run "dumbbell-topologyns3-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/.

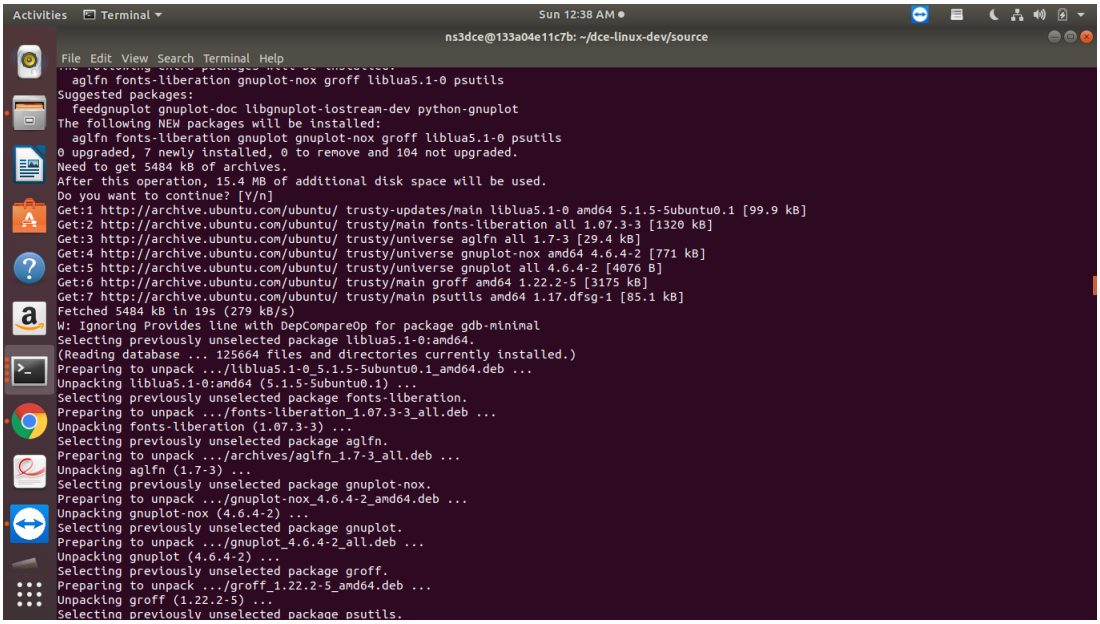


14. Under the latest timestamp folder, you will find a folder "cwndTraces" where A-ns3.plotme file will be generated. Copy this file in overlapped folder.

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.



16. copy CwndA.png to Home.

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

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