D0020E

PR-SCTP over IEEE 802.11

Appendix

March 11, 2017

1. Introduction

This project aims to asses the performances of the *SCTP, DCCP, TCP* and *UDP* network transport-layer protocols during different scenarios and *Random Early Detection (RED)* Algorithm, using the network simulator *NS-3*. The *NS-3* framework *Direct Code Execution (DCE)* is used to simulate client/server applications which generate network traffic. For the *DCCP, TCP* and *UDP* protocols, the standard Linux kernel implementations are used, while the *SCTP* protocol uses the external library called *lksctp*.

2. Installation

Before you can run anything, you need to install the dependencies, NS-3 and DCE.

2.1. Dependency Libraries

Use the following command to get all libraries along with NS-3 and DCE has to be installed.

```
# Dependencies
sudo apt-get install gcc g++ python python-dev qt4-dev-tools libqt4-dev \
                     mercurial bzr cmake libc6-dev g++-multilib gsl-bin \
                     libgsl0-dev libgsl2 flex bison libfl-dev tcpdump sqlite \
                     sqlite3 libsqlite3-dev libxml2 libxml2-dev \
                     libgtk2.0-0 libgtk2.0-dev vtun lxc doxygen graphviz \
                     imagemagick git python-pygraphviz python-pygoocanvas \
                     libgoocanvas-dev python-setuptools libpcap-dev libdb-dev \
                     libssl-dev lksctp-tools libsctp-dev tshark gnuplot cvs \
                     unrar p7zip-full autoconf
```

2.2. NS-3 with DCE

Run the script below to install NS-3 with DCE.

```
# !/bin/bash

cd ~

# Dependencies
sudo apt-get install gcc g++ python python-dev qt4-dev-tools libqt4-dev \
                     mercurial bzr cmake libc6-dev g++-multilib gsl-bin \
                     libgsl0-dev libgsl2 flex bison libfl-dev tcpdump sqlite \
                     sqlite3 libsqlite3-dev libxml2 libxml2-dev \
                     libgtk2.0-0 libgtk2.0-dev vtun lxc doxygen graphviz \
                     imagemagick git python-pygraphviz python-pygoocanvas \
                     libgoocanvas-dev python-setuptools libpcap-dev libdb-dev \
                     libssl-dev lksctp-tools libsctp-dev tshark gnuplot cvs \
                     unrar p7zip-full autoconf

# Installing the bake tool in the directory you're currently in.
hg clone http://code.nsnam.org/bake bake
export BAKE_HOME=`pwd`/bake
export PATH=$PATH:$BAKE_HOME
export PYTHONPATH=$PYTHONPATH:$BAKE_HOME
```

```
# Create the directory in which to download and build the DCE variant of NS-3.
# It has only been tried with version 1.7. Try another version at your own risk.
mkdir ns3-dce
cd ns3-dce
bake.py configure -e dce-linux-1.7
bake.py download

sed -i '75s/.*//' source/ns-3-dce/test/test-tsearch.cc

bake.py build
```

If there are a problem with installing the dce-linux-1.7. You could try doing this (as some computers have a problem with the python code in the installation script):

CARL STUFF PYTHON STUFF

The original installation instructions from the NS-3 webpage.

2.3. System settings

Most Linux systems place restrictions on how many user processes can be run at the same time, and how many files each process can open. This project needs to run multiple NS-3 simulation instances in order to generate useful network statistics, which creates a lot of files and processes. Therefore it is necessary to append the following lines to the end of

```
/etc/security/limits.conf
```

```
*         hard    nproc       65536
*         soft    nproc       65536
*         hard    nofile      65536
*         soft    nofile      65536
```

Restart your computer to apply the settings. For more information see the DCE Manual.

2.4. Downloading the project

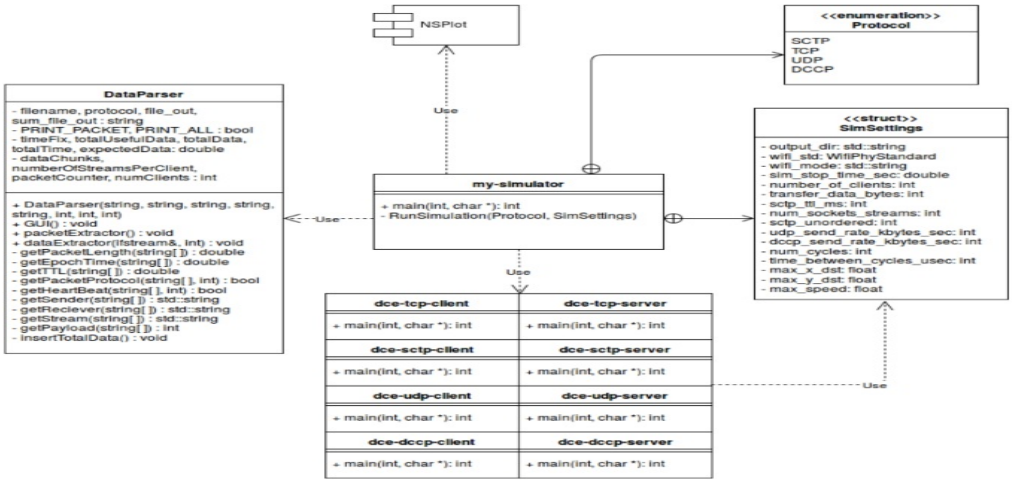
Assuming that the NS-3 DCE installation directory is:

```
export NS3_HOME="$HOME/dce"
```

```
cd $NS3_HOME/source/ns-3-dce
git init
git remote add origin https://github.com/Kininia/Network-Protocol-Simulation.git
git fetch
git checkout -t origin/master
```

3. Running the simulator

The simulator is built to run in the system terminal. The simulator first runs a simulation for each of the network protocols implemented then it parses the data from the dumped .pcap files into .dat files for the plotting program. Each protocol implemented has both a client and a server class. A diagram of the entire simulator is shown in figure 1.



Running the simulator is done through the terminal.

3.1. Starting

To run the simulator, run the following commands

```
./waf configure --with-ns3=$NS3_HOME/build --prefix=$NS3_HOME/build \
                --enable-kernel-stack=$NS3_HOME/source/net-next-sim-2.6.36/arch
./waf build
./waf --run my-simulator
```

3.2. Plotting

The plotting program is separate from the simulation. To plot the results from a simulation, run:

```
cd $NS3_HOME/source/ns-3-dce
build/myscripts/my-simulator/bin/./NSplot output.png my-simulator-output/*.dat -2d
```

3.3. Configuration Simulation Parameters

There are several premade scripts that are ready to be run. The scripts are named *SimRED* or *SimREG* depending on if it uses *RED* or not. Then they are named after which protocols they are running. For example *SimREG_All*, runs all of the scripts. While *SimRED_SCTP*, runs the *SCTP* protocol with *RED*. The different scripts that are premade are:

```
SimREG_All
SimREG_DCCP
SimREG_TCP
SimREG_UDP
SimREG_SCTP
SimRED_All
SimRED_DCCP
SimRED_TCP
SimRED_UDP
SimRED_SCTP
```

In order to change the parameters for the simulator the main section of the script has to be edited.

3.3.1. Wifi standard

The following lines has to be edited in order to change what Wifi standard the simulator uses.

```
sim settings.wifi std = WIFI PHY STANDARD 80211a;
sim settings.wifi mode = "OfdmRate54Mbps";
```

Currently there are five different Wifi standards implemented.

```
WIFI PHY STANDARD 80211a
WIFI PHY STANDARD 80211b
WIFI PHY STANDARD 80211g
WIFI PHY STANDARD 80211n 2 4 GHZ
WIFI PHY STANDARD 80211ac
```

There are also many different data rates implemented. Here are a few examples.

```
"OfdmRate6Mbps"
"OfdmRate9Mbps"
"OfdmRate12Mbps"
"OfdmRate18Mbps"
"OfdmRate24Mbps"
"OfdmRate36Mbps"
"OfdmRate48Mbps"
"OfdmRate54Mbps"
```

3.3.2. Simulation stop time

By changing the sim_settings.sim_stop_time_sec variable, one changes the max time the simulator is going to run. It is currently set to 10000 seconds.

3.3.3. Number of clients

Adding more clients to the simulator is done through changing the sim_settings.number_of_clients variable. The simulator will always have one server node in addition to the clients.

3.3.4. Transfer Bytes

Currently the amount of bytes to transfer from each client is 1 megabyte. This can be increased or decreased by changing the sim_settings.transfer_data_bytes variable.

3.3.5. Socket and Streams

The number of SCTP streams and the number of TCP & DCCP connections aswell as the number of UDP sockets are all configured by the sim_settings.num_sockets_streams variable.

3.3.6. SCTP

For SCTP both the time-to-live and unordered packet option can be configured. Changing the sim_settings.sctp_ttl_ms variable configures the time-to-live in milliseconds with 0 such that time-to-live will be disabled. Setting sim_settings.sctp_unordered variable to 0 sends the data in-order and setting it to 1 sends the data unordered.

3.3.7. UDP and DCCP

Both UDP and DCCP send rates can be configured with the

```
sim_settings.udp_send_rate_kbytes_sec
sim_settings.dccp_send_rate_kbytes_sec
```

variables. Both are measured in kilobytes/second.

3.3.8. Cycles

Configuring cycles changes if the simulator should run on/off sources or a single continuous stream. The sim settings.num cycles variable changes how many cycles is going to be run. Giving a total data sent per client = transfer data bytes * num cycles. The time between each cycle is configured by the sim_settings.time_between_cycles_usec variable and is measured in microseconds.

3.3.9. Output

The output directory for the simulator can be configured by the sim_settings.output_dir variable and is currently set to "/my-simulator-output/".

3.3.10. Variable Span

In order to change what variable span to iterate over the int *var variable has to be altered. It takes the memory address of the variable of iteration. One also has to change the int min, max, inc variables to fit the new iteration variable and if necessary change the loop itself.

4. NS-PLOT

4.1. Introduction

NS-PLOT is a module within the project *PR-SCTP over IEEE 802.11* which is responsible for the visual representation of the simulation result. *NS-PLOT* is a seperate program, which will create plots from the output of the *NS-3* simulation script. *NS-PLOT* takes one or more *.dat* files as input and output a single plot (represented as a *.png* file).

4.2. Use of Gnuplot

NS-PLOT uses *Gnuplot* for plotting. *Gnuplot* is a command-line program that can generate two- and three-dimensional plots of functions,data and data fits. Its frequently used for publication-quality graphics as well as education. The program runs on all major computers and operating systems (*GNU/Linux, Unix, Microsoft Windows, Mac OS X,* and others). IT is a program with a fairly long history, dating back to 1986.

NS-PLOT is handling the destination of input and output files, as well as the desired parameters, labels and type of graph (2D and 3D). This information is then passed to *Gnuplot* via a system call (**system()** function) which will generate an output plot.

4.3. Input files

The software takes at least one *.dat* file as input (limited to one file for 3D-plotting). These files contain the simulation summary. Each row in a *.dat* file stores a specific information about one simulation, for example, by looking at the figure below, we can see that the total amount of frames sent was 8129 it took approximately 5.03 seconds to transfer the total data of 10.73 Megabytes of data, of which 10 megabytes was the raw (actual/useful) data and so on. More information about the parameters can be found in subsection 4.3.

```
1628 2.58259 2.14592 2 2 93.2 0 0.83092 1382.16 2048 1024 1 4
3262 3.19647 4.2919 4 4 93.1989 0 1.3427 1379.64 4096 1024 1 4
4897 3.80911 6.43797 6 6 93.1971 0 1.69015 1378.54 6144 1024 1 4
6502 4.40991 8.58163 8 8 93.2224 0 1.94599 1383.96 8192 1024 1 4
8129 5.02668 10.727 10 10 93.2223 0 2.13402 1383.7 10240 1024 1 4
9758 5.63392 12.8726 12 12 93.221 0 2.28484 1383.27 12288 1024 1 4
11372 6.24717 15.017 14 14 93.2276 0 2.40381 1384.67 14336 1024 1 4
13032 6.86793 17.1651 16 16 93.2125 0 2.49931 1381.13 16384 1024 1 4
14643 7.46976 19.3092 18 18 93.2197 0 2.58498 1382.72 18432 1024 1 4
16272 8.0893 21.4548 20 20 93.2192 0 2.65224 1382.56 20480 1024 1 4
17876 8.70036 23.5984 22 22 93.2267 0 2.71235 1384.24 22528 1024 1 4
19551 9.32163 25.7477 24 24 93.2124 0 2.76214 1380.92 24576 1024 1 4
21137 9.91023 27.8898 26 26 93.2241 0 2.81424 1383.57 26624 1024 1 4
22767 10.5254 30.0355 28 28 93.2231 0 2.85362 1383.34 28672 1024 1 4
24389 11.1372 32.1805 30 30 93.2242 0 2.88945 1383.56 30720 1024 1 4
25981 11.7249 34.3231 32 32 93.2317 0 2.92736 1385.26 32768 1024 1 4
27677 12.3761 36.4741 34 34 93.2169 0 2.94713 1381.86 34816 1024 1 4
29314 12.9881 38.6203 36 36 93.2153 0 2.97351 1381.47 36864 1024 1 4
30915 13.5896 40.7636 38 38 93.2204 0 2.99962 1382.62 38912 1024 1 4
```

Those simulation files should also be formatted according to a predefined pattern. The parameters should be separated by a single line space, there should be equal amount of parameters in each row, every parameter should either be a real number or be assigned a *NaN* (Not-a-Number) value.

4.4. Functionality

The software supports both 2D- and 3D plotting of simulation data. The user can assign one of 13 available parameters to each axis. The list of parameters is displayed below:

- **Frames (id= 1)** - Number of frames (packets) sent during the transmission.
- **Transmission time (id = 2)** - Total transmission time in seconds (s).
- **Data with headers (id = 3)** - Total amount of data transferred (including header data) in Megabytes (Mb).
- **Data no headers (id = 4)** - Total amount of data transferred (header data not included) in Megabytes (Mb).
- **Data expected (id = 5)** - Expected amount of raw data to be sent in Megabytes (Mb).
- **Data percentage (id = 6)** - Percentage of raw data transferred.
- **Data loss percentage (id = 7)** - Percentage of data that has been lost during the transmission.
- **Transmission speed (id = 8)** - Average transmission speed in Megabytes per seconds (Mb/s).
- **Average frame size (id = 9)** - Average frame (packet) size for the transmission in bytes (b).
- **Data chunks (id = 10)** -Number of data chunks transmitted (*SCTP* only).
- **Average chunk size (id = 11)** - Average chunk size for the transmission (*SCTP* only).
- **Clients (id = 12)** - Number of clients which participated in the transmission
- **Streams/sockets per client (id = 13)** - Number of streams/sockets per client. Streams are specific for *SCTP* protocol.

NS-PLOT supports viewing of multiple simulations on the same data plot. This option is only available for 2D-plotting and can be useful for comparing simulation results to each other. The labels and keys are set automatically by the software, according to the chosen parameters

4.5. Usage of NS-PLOT

NS-PLOT is seperated from the *NS-3 DCE* simulation script. It is highly recomended that the *C++* code is compiled by using the *gcc* compiler with *-std=c++* flag. After the compilation the software can be started from the terminal by changing the directory to the one where the *NS-PLOT* executable file is stored. A new user of the software may use the command with the *-help* flag to get some basic information about the program, as described below.

```
./NSplot -help
```

4.5.1. Argument format

The input file(s) and output destination are passed on into the program by listing them in the bash command (a single space shall be added in between each of the files). The *-dim* flag should be given as the last argument in the command. The only two possible values for the *-dim* flag are *-2d* or *-3d*. The correct format of arguments can be seen below.

```
./NSplot output_file input_file_1 input_file_2 ... input_file_n -dim
```

An argument error can be generated if the format, given above, is not followed. File not found error will occur if one of the input files is non-existent. Dimension error may occur by either passing the wrong *-dim* flag, or by passing more than one input file from the terminal in 3D mode.

4.5.2. Creating a plot

In the example below, *SCTP* and *TCP* protocols were simulated by increasing the amount of sent data by 2 Megabytes at a time from 2 Megabytes to 100, by using our *NS-3* simulation script. The simulation results were stored in *sctp_simtotal.dat* for the simulation of *SCTP* and *tcp* simtotal.dat for the simulation of *TCP* protocol.

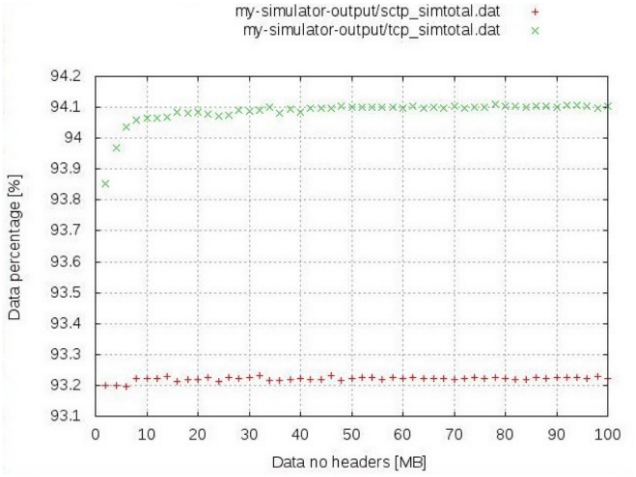
Let us define a case, where user wants to study how the percentage of useful data is affected by increasing the size of the file to transfer across a wireless link. The output should be a two-dimensional plot, stored in perc size.png file which will be located in the same folder. The correct command for starting up *NS-PLOT* will look as follows.

```
./NSplot perc_size.png sctp_simtotal.dat tcp_simtotal.dat -2d
```

Once the correct arguments are passed to the program and it starts running, user gets to choose the parameters which will be plotted against each other. The parameter which corresponds to the size of the file which was transferred is *Data no headers* parameter. This parameter has identification number 4, so user shall type number 4 to assign it to *x*-axis. The dependent parameter, which in this example is the percentage of useful/raw data, corresponds to *Data percentage* parameter. This parameter has identification number 6, and is assigned to the *y*-axis by typing 6 in to the program. The process is also described i figure 5

```
ns3@ubuntu:~/Desktop/NSPLOT-3.0$ ./NSplot perc_size.png sctp_simtotal.dat tcp_simtotal.dat -2d
NS-PLOT (v3.1)
********************************************************************************
Available parameters:
[1] Frames                  [2] Transmission time        [3] Data with headers
[4] Data no headers         [5] Data expected            [6] Data percentage
[7] Data loss percentage    [8] Transmission speed       [9] Avg. frame size
[10] Data chunks            [11] Avg. chunk size         [12] Clients
[13] Streams/sockets per client
********************************************************************************
Insert the first column of data that will be the x-axis : 4
[Data no headers] is selected for x-axis
Insert the second column of data that will be the y-axis : 6
[Data percentage] is selected for y-axis
********************************************************************************
Files for processing:
sctp_simtotal.dat.........valid
tcp_simtotal.dat.........valid
********************************************************************************
Setting Gnuplot parameters...
Number of dimensions: 2
Output PNG file: perc_size.png
********************************************************************************
```

Once the parameters are typed in correctly, *NS-PLOT* will generate an output, which will appear on the screen, as seen in the picture below.



4.5.3. Running NS-PLOT within the project

NS-PLOT is included in the project's *GitHub* repository and is compiled along with other libraries and helper-classes. Once all the components are installed and the user is able to perform simulations, the *NS-PLOT* module can be initiated by moving into the directory with the executable file.

```
cd dce/source/ns-3-dce
```

And running the following command to start the software with all simulation *.dat* files as input

```
build/myscripts/my-simulator/bin/./NSplot output.png my-simulator-output/*.dat -2d
```

Now, that the software is running, one can follow the steps provided in section 4.5.2 to create plots of the simulation data.