

---

# **ns-3 Direct Code Execution (DCE)**

## **Quagga Manual**

*Release 1.1*

**Direct Code Execution project**

August 30, 2013



# CONTENTS

|          |                                              |           |
|----------|----------------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                          | <b>3</b>  |
| 1.1      | Current Status (2012/4/23) . . . . .         | 3         |
| <b>2</b> | <b>Getting Started</b>                       | <b>5</b>  |
| 2.1      | Prerequisite . . . . .                       | 5         |
| 2.2      | Building ns-3, DCE, and DCE-Quagga . . . . . | 5         |
| 2.3      | Examples . . . . .                           | 6         |
| 2.4      | Configuration Manual . . . . .               | 7         |
| <b>3</b> | <b>Modifying DCE Quagga</b>                  | <b>9</b>  |
| 3.1      | Customizing Helper . . . . .                 | 9         |
| 3.2      | Customizing Binary . . . . .                 | 9         |
| <b>4</b> | <b>FAQ</b>                                   | <b>11</b> |



Contents:



# INTRODUCTION

The Quagga support on DCE enables the users to reuse routing protocol implementations of Quagga (RIPv1, RIPv2, RIPv3, OSPFv2, OSPFv3, BGP, BGP+, RAadv) as models of network simulation. It reduces the time of re-implementation of the model, and potentially improve the result of the simulation since it already “actively running” in the real world.

It was started as a Google Summer of Code (GSoC) 2008 by Liu Jian with numerous contributions especially on the netlink implementation with quagga porting into ns-3-simu (former name of ns-3-dce). You can look at his effort at the [link](#).

## 1.1 Current Status (2012/4/23)

Quagga support on DCE does not fully support all the environment that DCE has. The following shows the limited availability of each protocol.

|                 | Basic Mode (ns-3 stack) | Advanced Mode (ns-3-linux) | Remarks     |
|-----------------|-------------------------|----------------------------|-------------|
| Rtadvd (zebra)  | NG                      | OK                         |             |
| RIPv1/v2 (ripd) | NG                      | OK                         | bind() fail |
| RIPv3 (ripngd)  | NG                      | OK                         | send() fail |
| OSPFv2 (ospfd)  | OK                      | OK                         |             |
| OSPFv3 (ospf6d) | NG                      | OK                         | send() fail |
| BGP (bgpd)      | OK                      | OK                         |             |
| BGP+ (bgpd)     | NG                      | OK                         |             |



# GETTING STARTED

## 2.1 Prerequisite

Quagga support on DCE requires several packages: autoconf, automake, flex, git-core, wget, g++, libc-dbg, bison, indent, pkgconfig, libssl-dev, libsysfs-dev, gawk

You need to install the correspondent packages in advance.

```
$ sudo apt-get install git-core (in ubuntu/debian)
```

or

```
$ sudo yum install git (in fedora)
```

## 2.2 Building ns-3, DCE, and DCE-Quagga

To install ns-3-dce-quagga, you can use **bake** as an installation tool as follows.

```
$ hg clone http://code.nsnam.org/bake bake
$ export BAKE_HOME=`pwd`/bake
$ export PATH=$PATH:$BAKE_HOME
$ export PYTHONPATH=$PYTHONPATH:$BAKE_HOME
```

then build ns-3-dce with quagga:

```
mkdir dce
cd dce
bake.py configure -e dce-ns3-|version| -e dce-quagga-|version|
bake.py download
bake.py build
```

note that dce-quagga-1.1 is the DCE quagga module version 1.1. If you would like to use the development version of the module, you can specify **dce-quagga-dev** as a module name for bake.

If you want to use dce-quagga with DCE advanced mode (i.e., using Linux native network stack), you can build as following commands. This is highly recommended at this moment (2012/04/20) so that Quagga runs successfully.

```
mkdir dce
cd dce
bake.py configure -e dce-linux-|version| -e dce-quagga-|version|
bake.py download
bake.py build
```

For more information about ns-3-dce core, please refer the DCE manual.

Then you can try an example of ns-3-dce-quagga as follows:

```
$ cd source/ns-3-dce
$ ./test.py -s dce-quagga
...
PASS: TestSuite dce-quagga 9.775 s
1 of 1 tests passed (1 passed, 0 skipped, 0 failed, 0 crashed, 0 valgrind errors)
```

You can see the above PASSED test if everything goes fine. Congrats!

## 2.3 Examples

### 2.3.1 Basic

```
$ cd source/ns-3-dce
$ ./waf --run dce-zebra-simple
```

if everything goes fine, you would see the file “routes.log” in the current directory as follows. The routes “10.1.0.0/24” and “10.2.0.0/24” was announced by ospfd accordingly.

```
Time: 70s
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
127.0.0.0         0.0.0.0          255.0.0.0        U        0      -     -    0
10.1.1.0          0.0.0.0          255.255.255.0    U        0      -     -    1
10.2.1.0          0.0.0.0          255.255.255.0    U        0      -     -    2
10.1.0.0          10.2.1.1         255.255.255.0    UGS     20     -     -    2
10.2.0.0          10.1.1.1         255.255.255.0    UGS     20     -     -    1
```

### 2.3.2 OSPF

Another example of OSPF is generating pcap file.

```
$ cd source/ns-3-dce
$ ./waf --run dce-quagga-ospfd
```

You would see the following parsed output by tcpdump.

```
$ tcpdump -r dce-quagga-ospfd-0-0.pcap -n -vvv
:
(snip)
09:00:45.106325 IP (tos 0x0, ttl 1, id 0, offset 0, flags [none], proto OSPF (89), length 72, bad ck
  10.0.0.2 > 10.0.0.1: OSPFv2, Database Description, length 52
    Router-ID 10.0.0.2, Backbone Area, Authentication Type: none (0)
    Options [External], DD Flags [Master], MTU: 65535, Sequence: 0x4b3d3b2e
    Advertising Router 10.0.0.2, seq 0x80000002, age 0s, length 16
      Router LSA (1), LSA-ID: 10.0.0.2
      Options: [External]
```

### 2.3.3 OSPF with ns-3-linux

The final example of OSPF is using Linux kernel stack via DCE.

```
$ cd source/ns-3-dce
$ ./waf --run "dce-quagga-ospfd --netStack=linux"
```

then, you would see the following parsed output by tcpdump.

```
$ tcpdump -r dce-quagga-ospfd-0-0.pcap -n -vvv
:
(snip)
09:00:45.106325 IP (tos 0xc0, ttl 1, id 15116, offset 0, flags [none], proto OSPF (89), length 72)
  10.0.0.2 > 10.0.0.1: OSPFv2, Database Description, length 52
    Router-ID 10.0.0.2, Backbone Area, Authentication Type: none (0)
    Options [External], DD Flags [Master], MTU: 1500, Sequence: 0x4b3d3b2e
    Advertising Router 10.0.0.2, seq 0x80000002, age 0s, length 16
      Router LSA (1), LSA-ID: 10.0.0.2
      Options: [External]
```

## 2.4 Configuration Manual

In order to utilize quagga protocols in ns-3, users need to define in the scenario via ns3::QuaggaHelper.

```
#include "ns3/quagga-helper.h"

int main (int argc, char *argv[])
{
    QuaggaHelper quagga;
    quagga.EnableOspf (node, "10.0.0.0/8");
    quagga.EnableOspfDebug (node);
    quagga.EnableZebraDebug (node);
    quagga.Install (node);
}
```



# MODIFYING DCE QUAGGA

## 3.1 Customizing Helper

At this moment, only a limited configuration of Quagga is implemented in the QuaggaHelper. For example, if you wanna configure the “cost” parameter of OSPF link, you do have to extend QuaggaHelper (quagga-helper.cc) to generate the following configuration for example.

```
interface sim0
    ip ospf cost 20
!
```

## 3.2 Customizing Binary

If you wanna extend the protocol by modifying the source code of Quagga, your extended binary should be located at the directory “ns-3-dce/build/bin\_dce”.



## FAQ

(TBA)