**Step by Step for Testing - Handbook**

To carry out the tests, scripts were developed to automate the steps necessary to execute the tests. However, the tests can be performed manually as per step by step below.

The macro process of working the solution:

1. Texto

   Descrição gerada automaticamenteInterface gráfica do usuário, Texto

   Descrição gerada automaticamenteStart mininet via “Make run” command. Before executing this, it is always recommended to execute “make stop” and “make clean”.
2. Within the mininet open xterm of the destination host, example “ xterm h1c3”. Use XMING client .

Tela preta com letras brancas

Descrição gerada automaticamente

1. Run on mininet open xterm of the source host, example “ xterm h1c1”. Use XMING client .

Texto

Descrição gerada automaticamente

1. Start the receive.py script on the destination host, for example “./ receive.py infarr0.log infarr0”

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

1. Start the send.py script on the source host, for example “./ send.py 10.3.1.1 760 500”, where parameter 760 is the number of packages to be sent and parameter 500 refers to the size of the packages

Texto

Descrição gerada automaticamente

1. Run script that simulates link failure. In another terminal it is necessary to access the mininet registers console to configure that the port is down . It is necessary to identify the connection port of each switch. In the example below we will be simulating the DOWN port between S1CORE and A1P3. It is necessary to place the S1CORE port and the A1P3 port, so two interactions are required.

Commands:

simple\_switch\_ CLI -- thrift-port 9094 < port3down.txt

simple\_switch\_ CLI -- thrift-port 9102 < port3down.txt

Texto

Descrição gerada automaticamente

The identification of the connection port of the switches and the respective connection port for fault simulation were optimized through the program that was used during the automation of the test preparation process. The table with the connection port of each switch is available in the file port\_comutador.txt.

1. Run script that simulates link recovery. Similar to the process of putting the port in DOWN, a process must be performed to reactivate it.

Commands:

simple\_switch\_ CLI -- thrift-port 9094 < port3up.txt

simple\_switch\_ CLI -- thrift-port 9102 < port3up.txt

**Texto

Descrição gerada automaticamente**

**Step by step for preparing the test environment - SCRIPT**

All the steps described above are automated through the python algorithm “testatodos.py”. This algorithm has the ability to coordinate the execution of all necessary combinations of tests between hosts or to prepare the execution of a specific test.

Different versions and distributions of Linux, Mininet /, Python or the P4 compiler itself may require adaptations to the structure of variables and the return of commands. All work was done on an image distributed to a P4 community event on 2019-08-15.

Texto

Descrição gerada automaticamente

To automatically run xterm on mininet , update the file with the information below, updated file available on github .

1. Configure run\_exercise.py

/home/p4/tutorials/utils/run\_exercise.py

vi ~/tutorials/utils/run\_exercise.py

The parameters below must be adjusted

Texto

Descrição gerada automaticamente

1. Configure Makefile

/home/p4/FRRFTK4/ Makefile

Texto

Descrição gerada automaticamente

1. Configure startup script

sudo vi ~ /. bashrc

include in the last line ./runxterm.sh

Texto

Descrição gerada automaticamente

To run the tests, use the following parameters

./ testatodos.py < namealgorithm > <destination> <numbererrors> \_ \_

- Algorithm name : Infarr, rotor, static or plane control

- Destination : 1 for all tests or for specific tests you can determine the source host/ pod and destination host/ pod < idhostsource >< idpodsource >< idhostdest >< idpoddest >

- Error number : 0 - no error, 1 - for one error, 3 - for two errors, 4 - for three errors

Example:

1. To run tests of the INFARR algorithm for all hosts simulating a failure

./ testatodos.py infarr 1 1

1. To run tests of the INFARR algorithm between the H1C1 and H1C3 simulating a failure

./ testatodos.py infarr 1113 1

The lab environment was automated so that mininet automatically opens the xterms of the hosts involved and a host initialization script interprets whether the host should execute the receive.py or send.py file automatically.

In the laboratory environment, the errors in the links were programmed to occur in packet 250, in order to enable the fast rerouting behavior of the network in a controlled manner.

**Running Test Script**

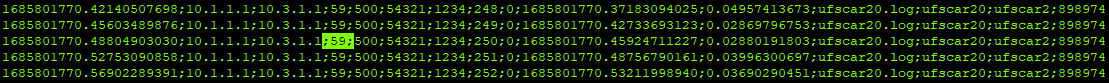
Running and interpreting the logs, we reinforce that the simulated error occurs in package 250:

1. INFAAR error logs, NO ERRORS.

./ testatodos.py infarr 1113 0

Highlighted field shows the TTL number of each packet. It is observed that the number of TTL does not change since there is no error.

infarr 0 .log file will be generated

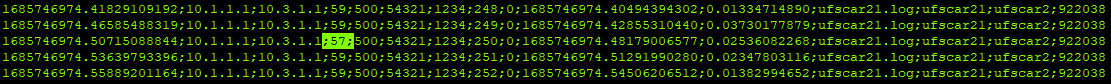
cat infarr 0 .log

1. INFARR error logs, 1 ERROR.

./ testatodos.py infarr 1113 1

infarr 1 .log file will be generated

Highlighted field shows the number of TTL sent to each packet. It is observed that the communication between packets is changed to 57 only during the recovery process (the failure occurs with an increase in the number of hops) (the lower the TTL, the more the number of hops). It is also observed that in the next packet the sending of packets returns to 59 TTL, thanks to the routing optimization promoted by INFFAR.

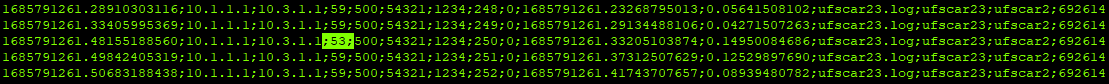
cat infarr 1 .log

1. INFARR error logs, 2 ERROR.

./ testatodos.py infarr 1113 two

infarr 2 .log file will be generated

Highlighted field shows the number of TTL sent to each packet. It is observed that the communication between packets is changed to 53 only during the recovery process (the failure occurs with an increase in the number of hops) (the lower the TTL, the more the number of hops). It is also observed that in the next packet the sending of packets returns to 59 TTL, thanks to the routing optimization promoted by INFFAR.

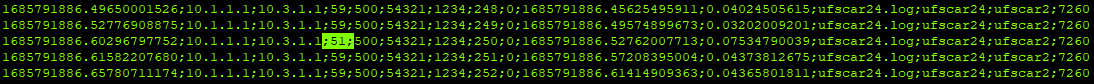
cat infarr 2 .log

1. INFARR error logs, 3 ERROR.

./ testatodos.py infarr 1113 3

infarr 3 .log file will be generated

Highlighted field shows the number of TTL sent to each packet. It is observed that the communication between packets is changed to 51 only during the recovery process (the failure occurs with an increase in the number of hops) (the smaller the TTL, the more the number of hops). It is also observed that in the next packet the sending of packets returns to 59 TTL, thanks to the routing optimization promoted by INFFAR.

cat infarr 3 .log

The package when sent by the HOST starts with TTL=64, with each routing performed there is a decrease in the number of TTL.