**Passo a Passo para realização dos testes - Manual**

Para realização dos testes foram desenvolvidos scripts para automação das etapas necessárias para execução dos testes. Todavia os testes podem ser realizados manualmente conforme passo a passo abaixo.

O processo macro de funcionando da solução:

1. Texto

   Descrição gerada automaticamenteInterface gráfica do usuário, Texto

   Descrição gerada automaticamenteIniciar mininet através comando “Make run”. Antes da execução deste recomenda-se sempre a execução do “make stop” e “make clean”.
2. Dentro do mininet abrir xterm do host destino, exemplo “xterm h1c3”. Use XMING client.

Tela preta com letras brancas

Descrição gerada automaticamente

1. Dentro do mininet abrir xterm do host origem, exemplo “xterm h1c1”. Use XMING client.

Texto

Descrição gerada automaticamente

1. Iniciar no host destino o script receive.py, exemplo “./receive.py infarr0.log infarr0”

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

1. Iniciar no host origem o script send.py, exemplo “./send.py 10.3.1.1 760 500”, onde o parâmetro 760 são os números de pacotes que serão enviados e o parâmetro 500 se refere ao tamanho dos pacotes

Texto

Descrição gerada automaticamente

1. Executar script que simula falha no enlace. Em outro terminal é necessário acessar a console os registradores do mininet para configurar que a porta esta down. É necessário identificar a porta de conexão de cada comutador. No exemplo abaixo estaremos simulando a porta DOWN entre o S1CORE e A1P3. É necessário colocar a porta do S1CORE e a porta do A1P3, assim são necessárias duas interações.

Comandos:

simple\_switch\_CLI --thrift-port 9094 < porta3down.txt

simple\_switch\_CLI --thrift-port 9102 < porta3down.txt

Texto

Descrição gerada automaticamente

A identificação da porta de conexão dos comutadores e a respectiva porta de conexão para simulação da falha foram otimizadas através do programa que foi utilizado durante a automação do processo de elaboração de testes. A tabela com a porta de conexão de cada comutador esta disponivel no arquivo port\_comutador.txt.

1. Executar script que simula recuperação do enlace. Similar ao processo de colocar a porta em DOWN, deve-se executar um processo para reativá-lo.

Comandos:

simple\_switch\_CLI --thrift-port 9094 < porta3up.txt

simple\_switch\_CLI --thrift-port 9102 < porta3up.txt

**Texto

Descrição gerada automaticamente**

**Passo a Passo para preparação do ambiente de testes - SCRIPT**

Todos as etapas descritas acima são automatizadas através do algoritmo em python “testatodos.py”. Este algoritmo possui a habilidade de coordenar a execução de todas as combinações necessárias de testes entre os hosts ou de preparar a execução de um teste específico.

Diferentes versões e distribuições do Linux, Mininet /, Python ou do próprio compilador P4 podem requerer adaptações diante a estrutura de variáveis e retorno dos comandos. Todo trabalho foi realizado sobre uma imagem distribuída para um evento da comunidade P4 em 2019-08-15.

Texto

Descrição gerada automaticamente

Para execução automática do xterm no mininet atualize o arquivo com as informações abaixo, arquivo atualizado disponivel no github.

1. Configure run\_exercise.py

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

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

O parâmetros abaixo devem ser ajustados

Texto

Descrição gerada automaticamente

1. Configure Makefile

/home/p4/FRRFTK4/Makefile

Texto

Descrição gerada automaticamente

1. Configure script inicialização

sudo vi ~/.bashrc

incluir na ultima linha ./runxterm.sh

Texto

Descrição gerada automaticamente

Para execução dos testes deve-se se usar os seguintes parâmetros

./testatodos.py <nomealgoritimo> <destino> <numeroerros>

- Nome do algoritmo: Infarr, rotor, estático ou planocontrole

- Destino: 1 para todos os testes ou para testes específicos pode-se determinas o host/pod origem e host/pod destino <idhostsource><idpodsource><idhostdest><idpoddest>

- Número erro: 0 - sem erro, 1 – para um erro, 3 – para dois erros, 4 – para três erros

Exemplos:

1. Para executar testes do algoritmo INFARR para todos os hosts simulando uma falha

./testatodos.py infarr 1 1

1. Para executar testes do algoritmo INFARR entre o H1C1 e H1C3 simulando uma falha

./testatodos.py infarr 1113 1

O ambiente de laboratório foi automatizado para que automaticamente o mininet abra os xterm dos hosts envolvidos e um script de inicialização do host interpreta se o host deve executar o arquivo receive.py ou send.py automaticamente.

No ambiente de laboratório os erros nos enlaces foram programados para ocorrem no pacote 250, de forma viabilizar de maneira controlado o comportamento do reroteamento rápido da rede.

**Executando Script de testes**

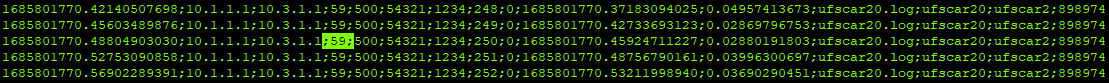
Executando e Interpretando os logs, reforçamos que o erro simulado ocorre no pacote 250:

1. Logs de erro INFAAR, SEM ERROS.

./testatodos.py infarr 1113 0

Campo destacado mostra o número de TTL de cada pacote. Observa-se que o número de TTL não sobre alteração já que não ocorrência de erro.

Será gerado o arquivo infarr0.log

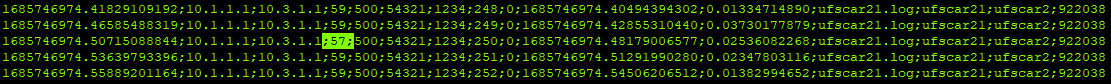
cat infarr0.log 

1. Logs de erro INFARR, 1 ERRO.

./testatodos.py infarr 1113 1

Será gerado o arquivo infarr1.log

Campo destacado mostra o número de TTL enviados a cada pacote. Observa-se que a comunicação entre os pacotes é alterada para 57 somente durante o processo de recuperação (ocorre a falha há um aumento no número de saltos) (quanto menor o TTL, mais será o número de saltos). Observa-se ainda que no pacote seguinte o envio de pacotes retorna a 59 TTL, graças a otimização do roteamento promovido pelo INFFAR.

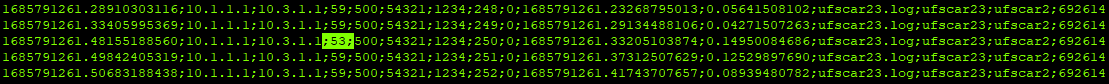
cat infarr1.log

1. Logs de erro INFARR, 2 ERRO.

./testatodos.py infarr 1113 2

Será gerado o arquivo infarr2.log

Campo destacado mostra o número de TTL enviados a cada pacote. Observa-se que a comunicação entre os pacotes é alterada para 53 somente durante o processo de recuperação (ocorre a falha há um aumento no número de saltos) (quanto menor o TTL, mais será o número de saltos). Observa-se ainda que no pacote seguinte o envio de pacotes retorna a 59 TTL, graças a otimização do roteamento promovido pelo INFFAR.

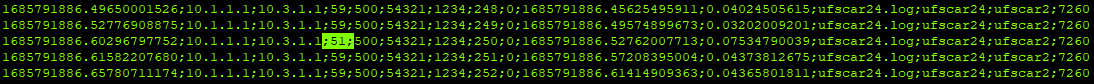
cat infarr2.log

1. Logs de erro INFARR, 3 ERRO.

./testatodos.py infarr 1113 3

Será gerado o arquivo infarr3.log

Campo destacado mostra o número de TTL enviados a cada pacote. Observa-se que a comunicação entre os pacotes é alterada para 51 somente durante o processo de recuperação (ocorre a falha há um aumento no número de saltos) (quanto menor o TTL, mais será o número de saltos). Observa-se ainda que no pacote seguinte o envio de pacotes retorna a 59 TTL, graças a otimização do roteamento promovido pelo INFFAR.

cat infarr3.log

O pacote quando enviado pelo HOST começa com o TTL=64, a cada roteamento realizado ocorre um decremento do número de TTL.