INSTITUTO FEDERAL
Santa Catarina
Câmpus São José

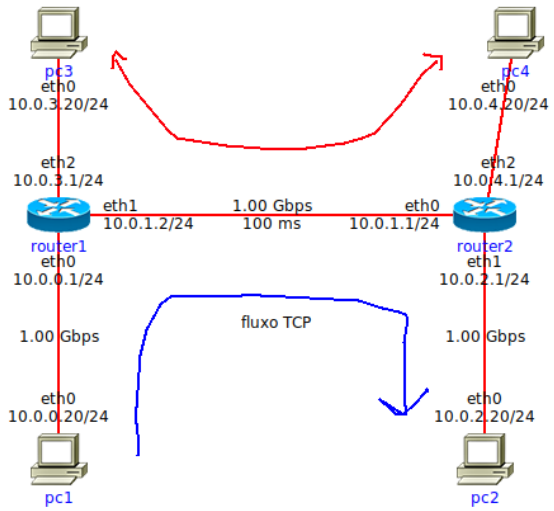
Avaliação de Desempenho de Sistemas

Mini Projeto ADS - Avaliação de Desempenho do Protocolo TCP em condições de congestionamento: cubic x reno

Objetivo

O protocolo TCP possui mecanismos de controle de congestionamento, além dos mecanismos de controle de erro e de fluxo. Projete um experimento para avaliar o comportamento de dois algoritmos de controle de congestionamento (cubic e reno) em uma dada topologia (ver abaixo). Analise os resultados. Considere um projeto de experimento $2^k r$ onde:

- tráfego de background (médio, alto). A ser definido no experimento.
- Tráfego de background somente UDP.
- cenário com um fluxo TCP;
- algoritmo de congestionamento: cubic e reno;
- links são sempre ethernet 1Gbps. Considere todos links com BER 0 mas no link entre roteadores com BER 1/1000000 e BER 1/100000;
- retardo forçado no link entre roteadores: 10ms e 100ms;
- usar imunes e IPERF;
- usar janela default do TCP;
- comparar usando gráficos de barras mostrando intervalo de confiança de 99%;
- elaborar um pequeno relatório seguindo os pontos analisados no artigo repassado como tarefa A1.1.
- automatizar, se possível a parte de coleta e de geração de gráficos. Ver anexo.



INSTITUTO FEDERAL
Santa Catarina
Câmpus São José

Automação do experimento: processo de coleta e dados

Uso de ferramentas do Imunes

Para a automação da execução do experimento recomenda-se usar as ferramentas disponíveis no imunes (ver wiki do imunes):

- himage:comando para execução de comando na máquina do cenário simulado a partir da máquina hospedeira;
- vlink: comando para mudar configurações de link no cenário simulado a partir da máquina hospedeira.
- hcp: comando para cópia de arquivos entre máquinas simuladas e hospedeiro.

Exemplo 1: executar o iperf servidor na máquina pc2:

sudo himage pc2@i2520 iperf -s

O identificador i2520 identifica o ID do experimento (pode ser visto na tela do Imunes, na barra de baixo.

Exemplo 2: mudar a configuração BER (Bit Error Rate) do link router2:pc2

sudo vlink -BER 1000000 router2:pc2@i2520

Notar que a modificação da condiguração do link não aparece na interface gráfica do imunes.

Uso de script python

Como será necessário repetir o experimento para 8 configurações diferentes, recomenda-se a construção de um script python. Exemplo de um possível script python para execução do iperf servidor.

```
import subprocess
alg = ['cubic', 'reno']
cmd_iperf_server=”sudo himage pc2@i2520 iperf -s -y C > dados.csv”
subprocess.run(cmd_iperf_server, shell=True,capture_output=True)
```

Notar que os dados foram armazenados em um arquivo de dados.csv. Alternativas para facilitar o posterior processamento seria:

1) Armazenar os dados em arquivos que identifiquem o experimento:
   Exemplo> dados-CUBIC-BER100000-r0
   Onde r0 identifica a replicação 0 do experimento
2) Armazenar todos os dados em um arquivo com append. Neste caso, a identificação do experimento deve ser acrescentada em cada linha de saída para facilitar o posterior processamento.

Automação do experimento: processo de geração de dados

Os dados gerados serão usado para geração de gráficos. Uma possibilidade é o uso do Jupiter para em um Codespace do github. Existe um modelo pronto. Exemplo de geração de gráfico de barras com python e matplotlib pode ser visto aqui.

Referências

- Site do Iperf
- Wiki do Imunes
- Slides sobre gráficos do Prof.Jain
- Slides sobre Intervalo de Confiança (aulas de ADS e referências dos slides)

ESQUELETO DE REFERÊNCIA

```
import subprocess

##### inicializacao de dados

alg = ['cubic', 'reno']
BER = ['100000', '1000000']
e2e_delay = ['10000', '100000']

repeticao = 8

cmd_iperf_client="sudo himage pc1@i3f30 iperf -c 10.0.0.21 -y C -Z " + alg[0] + " > dados-" + alg[0] + ".csv"

##### execucao dos experimentos

#para cada uma das repeticoes faca
for rep in range(repeticao):
        for proto in alg:
                for ber in BER:
                        for e2e in e2e_delay:
                                subprocess.run("sudo vlink -BER " + ber + "  pc1:pc2@i3f30 ", shell=True)
```

```
#configure o link usando o comando vlink

#execute o iperf cliente salvando os dados em arquivo separado (?)
#print(cmd_iperf_client)
subprocess.run(cmd_iperf_client, shell=True)
```