MAC0448/5910 - Programação para Redes de Computadores
EP3
Data de Entrega: 23/11/2014

Prof. Daniel Macêdo Batista

1 Problema

Neste EP você deverá implementar um simulador de redes que simule diversos computadores conversando entre si via rede. O simulador deverá simular protocolos das camadas de rede, transporte e aplicação e deverá permitir que qualquer enlace da rede seja monitorado, como se alguém tivesse colocado um sniffer naquele ponto e enxergasse todos os pacotes trafegando por ali em todos os sentidos.

Você também deverá fazer experimentos para comprovar o funcionamento do seu simulador. Assim como foi feito no EP2, neste EP você deverá entregar, além dos códigos, um .pdf com slides, **mesmo que você não vá apresentar**. Os slides deverão apresentar resultados dos experimentos mostrando o funcionamento do seu simulador.

2 Requisitos

2.1 Protocolos suportados

O seu simulador deverá simular os seguintes protocolos:

- Camada de rede: IP
- Camada de transporte: TCP e UDP
- Camada de aplicação: HTTP, FTP e DNS

2.2 Arquivo de entrada

O seu simulador deverá receber como entrada um arquivo descrevendo a simulação a ser realizada. O exemplo abaixo descreve o formato do arquivo. Ele é baseado no formato do arquivo de entrada do simulador de redes NS-2 [1], um dos simuladores de redes mais utilizados no meio acadêmico.

```
# Criação de 4 computadores
set h0 [$simulator host]
```

[1] `http://www.isi.edu/nsnam/ns/`

```
set h1 [$simulator host]
set h2 [$simulator host]
set h3 [$simulator host]

# Criação de 2 roteadores com 3 interfaces cada
set r0 [$simulator router 3]
set r1 [$simulator router 3]

# Criação dos enlaces ligando os computadores e os roteadores
# para criar a simulação abaixo:
#
#     h0 --------                    -------- h2
#                \                  /
#                 r0 ------------ r1
#                /                  \
#     h1 --------                    -------- h3
#
# O ’duplex-link’ diz que cada enlace tem capacidades em cada sentido
# independentes. Ou seja, se configuro o enlace com 2Mbps, significa que
# posso ter ao mesmo tempo 2Mbps fluindo num sentido e 2Mbps fluindo
# no outro sentido. O atraso configurado é igual nos dois sentidos. Ou
# seja, se configuro o link com 10ms significa que o atraso em um
# sentido é 10ms e o atraso no outro sentido é 10ms (Esse é o atraso
# sem contar com o enfileiramento nos roteadores e sem contar com o
# tempo para transmitir os bits em função da capacidade do enlace!)
# Obs.: roteadores tem várias portas, por isso é necessário colocar o
# .0, .1, .2 para o simulador saber em qual porta o enlace está
# conectado

$simulator duplex-link $h0 $r0.0 10Mbps 10ms
$simulator duplex-link $h1 $r0.1 5Mbps 2ms
$simulator duplex-link $r0.2 $r1.0 2Mbps 20ms
$simulator duplex-link $r1.1 $h2 10Mbps 5ms
$simulator duplex-link $r1.2 $h3 5Mbps 2ms

# Configuração dos hosts: endereço IP do computador, endereço IP do
# roteador padrão e endereço IP do servidor DNS
# Obs.: Considere que todas as redes sempre serão classe C, ou seja,
# máscara = 255.255.255.0
$simulator $h0 10.0.0.1 10.0.0.2 192.168.1.1
$simulator $h1 10.1.1.1 10.1.1.2 192.168.1.1
$simulator $h2 192.168.2.2 192.168.2.3 192.168.1.1
# Obs.: o endereço 1.1.1.1 representa um servidor DNS raiz que todos
# os servidores DNS da simulação precisam ter configurados como o
# servidor DNS deles (não vai ser usado para nada mas sempre tem que
# ter essa configuração)
$simulator $h3 192.168.1.1 192.168.1.2 1.1.1.1
```

```
# Configuração dos roteadores: porta, endereço IP [[porta, endereço
# IP] ...]
$simulator $r0 0 10.0.0.2 1 10.1.1.2 2 192.168.3.3
$simulator $r1 0 192.168.3.4 1 192.168.2.3 2 192.168.1.2

# Configuração dos roteadores: rotas (obs.: nos roteadores todas as
# rotas devem ser explícitas apontando para outro roteador ou para a
# porta. Não há roteador padrão no roteador. Não se preocupe com o
# caso de comunicações com endereços inexistentes na
# rede)
$simulator $r0 route 10.0.0.0 0 10.1.1.0 1 192.168.3.0 2 192.168.2.0 \
192.168.3.4 192.168.1.0 192.168.3.4
$simulator $r1 route 192.168.3.0 0 192.168.2.0 1 192.168.1.0 2 10.0.0.0 \
192.168.3.3 10.1.1.0 192.168.3.3

# Configuração dos roteadores: tempo para processar 1 pacote, porta,
# tamanho da fila da porta em quantidade de pacotes, [[porta, tamanho
# ...] ...]
$simulator $r0 performance 100us 0 1000 1 1000 2 1000
$simulator $r1 performance 20us 0 1000 1 1000 2 1000

# Configuração dos agentes da camada de aplicação. h0 e h1 vão rodar
# clientes HTTP. h2 vai ser um servidor HTTP e h3 vai ser o servidor DNS
# Obs.: não há necessidade de explicitar um cliente DNS. Todos os
# hosts já tem um cliente DNS embutido
# Obs.: no caso de FTP basta trocar tudo que tem HTTP ou http abaixo
# por FTP ou ftp
set httpc0 [new Agent/HTTPClient]
set httpc1 [new Agent/HTTPClient]
set https2 [new Agent/HTTPServer]
set dns3 [new Agent/DNSServer]
$simulator attach-agent $httpc0 $h0
$simulator attach-agent $httpc1 $h1
$simulator attach-agent $https2 $h2
$simulator attach-agent $dns3 $h3

# Configuração dos sniffers. Onde vou monitorar a rede. A saída tem que
# aparecer na saída padrão e nos arquivos configurados abaixo:
set sniffer1 [new Agent/Sniffer]
set sniffer2 [new Agent/Sniffer]
$simulator attach-agent $sniffer1 $r0.2 $r1.0 "/tmp/sniffer1"
$simulator attach-agent $sniffer2 $h1 $r0.1 "/tmp/sniffer2"

# Configuração das comunicações entre os agentes. Defino o instante de
# tempo em segundos quando o acesso deve ocorrer e o tipo de acesso a
# ser feito entre o cliente e o servidor. Note que 3 GETs do HTTP
```

```
# foram feitos com o nome da máquina, o que vai exigir um acesso DNS
# (considere que as máquinas não possuem cache.  Ou seja, elas
# sempre vão consultar um servidor DNS). Um dos acessos entretanto,
# está sendo feito direto pelo endereço IP, ou seja, nesse caso uma
# consulta DNS não deverá ser realizada. No caso de simulações com FTP
# é possível fazer tanto GET quanto PUT
$simulator at 0.5 "httpc0 GET h2"
$simulator at 0.6 "httpc1 GET 192.168.2.2"
$simulator at 0.6 "httpc1 GET h0"
$simulator at 0.7 "httpc0 GET h2"
$simulator at 4.0 "finish"
```

2.3 Arquivo de saída

A saída do simulador, que são as capturas feitas pelos sniffers, deve aparecer tanto na saída padrão quanto nos arquivos especificados para cada Agent/Sniffer. Não há um formato específico para os arquivos de saída. Eles devem estar em ASCII e apresentar de forma clara e fácil de visualizar as informações de cada pacote capturado. As informações devem estar separadas claramente para cada camada:

- identificador do pacote (um identificador único para cada pacote na simulação)
- instante de tempo em que o pacote foi visto (relógio da simulação. Não o relógio real)
- identificador do sniffer que pegou esse pacote
- Camada de rede (IP):

  Endereço IP de origem

  Endereço IP de destino

  Identificação do protocolo da camada de cima (o código correto segundo a RFC)

  Tamanho do cabeçalho IP + tamanho de tudo que vem nas camadas de cima

  TTL
- Camada de transporte (TCP):

  Porta fonte

  Porta destino

  Número de sequência

  Número de reconhecimento (Caso o bit ACK esteja ligado)

  Bit ACK

  Bit FIN

  Bit SYN
- Camada de transporte (UDP):

  Porta fonte

  Porta destino

  Tamanho do cabeçalho UDP + tamanho de tudo que vem na camada de cima

- Camada de aplicação (DNS):

  Pergunta feita ou resposta (em ASCII seguindo a RFC)

- Camada de aplicação (HTTP):

  Pergunta feita ou resposta (em ASCII seguindo a RFC)

- Camada de aplicação (FTP):

  Pergunta feita ou resposta (em ASCII seguindo a RFC)

2.4 Lógica do código

O tempo no simulador não deve ser considerado como o tempo real de relógio. Ou seja, se a simulação tiver poucos elementos e poucas comunicações, muito provavelmente 1 segundo no simulador será bem mais rápido do que 1 segundo de relógio. Entretanto, se a simulação tiver muitos elementos e muitas comunicações, muito provavelmente 1 segundo no simulador será mais lento do que 1 segundo de relógio.

Cada nó (computadores ou roteadores) na simulação deve ser uma entidade independente no seu código: uma instância de uma classe, uma thread, um processo, etc... Ou seja, seu simulador **não deve possuir uma entidade central que controla os envios e recepções de todos os pacotes**.

A simulação do protocolo HTTP deve mostrar as mensagens no nível da camada de aplicação que de fato seriam trocadas entre um cliente e um servidor HTTP. Você pode reproduzir as mensagens que seriam trocadas caso o servidor fosse o o Apache. Com relação ao conteúdo transferido entre o servidor e o cliente, você deve transferir um arquivo texto (ASCII) de no mínimo 500KB. Não ponha o conteúdo do arquivo hardcoded no seu código. Mantenha ele no mesmo diretório do seu simulador e vá lendo ele em tempo de execução sempre que uma simulação do GET for realizada. De forma similar deve ser feito com o FTP. Nesse caso você deve considerar que antes de qualquer transferência de dados o usuário logará com qualquer usuário e senha e que logo depois ele ou fará um GET para baixar um arquivo do servidor ou um PUT para enviar um arquivo para o servidor. As mensagens trocadas podem tanto seguir o que seria trocado caso o servidor fosse o seu servidor do EP1 ou caso o servidor fosse o proftpd.

Uma informação importante para simular o TCP corretamente é o MSS. O MSS é informado em um campo de Opções no cabeçalho do TCP no momento em que a conexão é estabelecida. No simulador considere que o MSS sempre será igual a 1460 bytes. Não há necessidade de simular a informação do MSS no momento de estabelecimento de conexão. Ainda sobre o TCP, note que o campo que informa o tamanho da janela de recepção não precisa ser informado, logo, não há necessidade de implementar o controle de fluxo. O controle de congestionamento, entretanto, precisa ser implementado.

Cuidado pois certos elementos no seu código representam entidades compartilhadas. Um enlace por exemplo não pode enviar dois pacotes ao mesmo tempo. O mesmo vale para as interfaces de entrada e saída dos roteadores e dos computadores. Note que não há limitação na quantidade de pacotes das interfaces dos computadores. Apenas na quantidade de pacotes nas interfaces dos roteadores. Caso esse limite seja alcançado, os pacotes devem ser descartados.

Mais uma dica: se você não lembra quais são os passos realizados em comunicações com os protocolos exigidos neste EP basta você rodar o wireshark na sua máquina e observar o tráfego de pacotes. Utilizando máquinas virtuais como a que você utilizou no EP2 você pode até mesmo criar cenários iguais aos simulados e observar o tráfego de pacotes neste ambiente virtual para verificar se seu simulador está se comportando como deveria.

2.5 Linguagem

Os programas podem ser escritos em qualquer linguagem de programação, desde que exista compilador gratuito para GNU/Linux, e devem funcionar no shell, sem interface gráfica. Certifique-se de que seu programa funciona no GNU/Linux pois ele será compilado e avaliado apenas neste sistema operacional.

Você não pode utilizar bibliotecas, classes ou similares que já implementem um simulador de redes. Códigos que não respeitem esse requisito terão nota ZERO.

2.6 Slides

Os seus slides devem ser feitos de modo a serem apresentados em um tempo máximo de 10 minutos. Os slides devem conter:

- Detalhes do ambiente onde os experimentos foram realizados tanto em termos de hardware quanto de software
- 1 gráfico em barras com o tempo de relógio e 1 gráfico em barras com a utilização da CPU do computador onde você rodou seu simulador em três cenários: um cenário simples, um cenário médio e um cenário complexo. Você deve repetir a simulação de cada cenário 30 vezes e apresentar nos gráficos em barra a média e o intervalo de confiança[2] para as medições.
- Comentários sobre os gráficos do item anterior que servem para avaliar o desempenho do seu simulador (os resultados encontrados foram os esperados?)
- Conclusões, focando nas dificuldades encontradas durante o desenvolvimento do EP e nos pontos positivos de ter realizado o EP, caso você ache que teve algum.
- Apresentação da saída do seu simulador para um cenário simples de modo a confirmar o correto funcionamento dos mecanismos de controle de congestionamento, estabelecimento de conexão e encerramento de conexão do TCP (Destaque na saída as informações que interessam. Não é para encher os slides com tudo que o simulador jogou na saída padrão)

3 Entrega

Você deverá entregar um arquivo .tar.gz contendo os seguintes itens:

- fonte do simulador;
- Makefile (ou similar);
- arquivo LEIAME;
- .pdf dos slides.

[2] Se você não lembra de intervalo de confiança das aulas de estatística, leia sobre isso no livro do Raj Jain (`http://www.amazon.com/The-Computer-Systems-Performance-Analysis/dp/0471503363`). A Biblioteca do IME tem exemplares desse livro.

O desempacotamento do arquivo .tar.gz deve produzir um diretório contendo os itens. O nome do diretório deve ser ep3-membros_da_equipe. Por exemplo: ep3-joao-maria.

A entrega do .tar.gz deve ser feita através do PACA.

O EP pode ser feito individualmente ou em dupla.

Obs.: Serão descontados pontos de EPs que não estejam nomeados como solicitado, que não criem o diretório com o nome correto após serem descompactados ou que não contenham todos os arquivos necessários.

Obs.: O prazo de entrega expira às 8:00:00 do dia 23/11/2014. EPs entregues com atraso terão -1,0 por cada hora de atraso. Por exemplo, se você entregar seu EP entre 8:00:01 e 8:59:59, ele valerá 9,0

4 Avaliação

60% da nota será dada pela implementação, 10% pelo LEIAME e 30% pelos slides, independente deles serem apresentados. Os critérios detalhados da correção serão disponibilizados apenas quando as notas forem liberadas. Aqueles que forem sorteados para arguição e apresentação terão como nota final do EP a média entre a nota do .tar.gz (distribuída como explicado no início do parágrafo), a nota da arguição e a nota da apresentação.