Redes de Computadores

TP1: transferência de arquivos usando janela deslizante
Data de entrega: verifique o calendário do curso
Conteúdo da entrega: `.tar.gz` ou `.zip` contendo um ou mais `.py` do código e um `.pdf` do relatório

Objetivos e etapas

O objetivo deste trabalho é (1) exercitar a interface de programação com sockets (em python), (2) se familiarizar com o ambiente virtualizado mininet para emulação de sistemas em rede e (3) medir e analisar o desempenho de aplicações simples neste ambiente. Para tanto, as seguintes etapas estão envolvidas:

1. Instalação do mininet e configuração de uma topologia.
2. Implementação/execução de programas de teste no mininet e medição de desempenho.
3. Análise de resultados e escrita do relatório.

Este trabalho tem por objetivo fazer com que os alunos experimentem na prática com as decisões de projeto necessárias para a implementação de protocolos de transmissão confiáveis sobre ambientes sujeitos a erros e perdas. Nesse processo, os alunos também deverão se se familiarizar com o ambiente virtualizado mininet[1] para emulação de sistemas em rede.

Dessa forma, o trabalho deve expor os alunos a pelo menos dois aspectos de projeto em redes de computadores:

- o uso de recursos de programação concorrente para organização de sistemas e
- os aspectos de implementação de um protocolo de entrega confiável com janela deslizante.

As seções a seguir descrevem o projeto em linhas gerais. Alguns detalhes são definidos, mas diversas decisões de funcionalidade, projeto e implementação estão a cargo dos alunos.

Instalação do mininet

Com esta seção, você irá entender o relacionamento da máquina hospedeira com a VM mininet e preparar o ambiente para configuração.

O mininet é um emulador de redes. Com ele é possível executar *hosts*, *switches* e controladores em um único kernel, virtualmente. Para utilizá-lo, você precisa de uma imagem do mininet (obtida aqui) e um sistema de virtualização instalado para executar essa imagem. A imagem do mininet é, basicamente, um kernel linux modificado e por isso recomendamos a última versão disponível e estável (Ubuntu 14.04 32 ou 64 bits) com o sistema de virtualização VirtualBox.

Para instalar o mininet basta criar uma VM com as especificações da imagem (Ubuntu 14.04/64 bits, por exemplo) e na etapa de configuração de disco, escolha a opção "usar um arquivo de disco existente"e selecione o arquivo de extensão `.vmdk` do mininet. Com isso você será capaz de inicializar a VM e se autenticar (usuário: `mininet`; senha: `mininet`).

Uma etapa adicional é tornar a sua VM acessível pela máquina hospedeira, permitindo acesso por `ssh` ou transferência de arquivos por `scp`. Para isso, com a VM desligada, vá em máquina:configurações:rede, escolha um novo adaptador de rede (adaptador 2, provavelmente) e adicione uma rede do tipo *host-only*.

Para que o SO da VM tome conhecimento e inicialize automaticamente a interface *host-only*, adicione-a no arquivo de interfaces (`/etc/network/interfaces`), ainda logado como usuário `mininet` (edite o arquivo com permissão de super-usuário `sudo`). O resultado é similar ao seguinte. Note que a interface NAT (para internet) e a *loopback* já existiam no arquivo.

```
auto lo # loopback
iface lo inet loopback

auto eth0 # host-only
iface eth0 inet dhcp

auto eth1 # NAT
iface eth1 inet dhcp
```

[1] http://mininet.org/

Siga os seguintes passos para verificar o sucesso de seu trabalho até aqui: (1) reinicialize a VM; (2) autentique-se como `mininet` e (3) execute `$ ifconfig -a`. Você deve visualizar as interfaces, inclusive a *host-only* (com endereço `192.168.56.102`, por exemplo). Se tudo estiver certo, você conseguirá abrir um shell seguro para a VM com `$ ssh mininet@192.168.56.102` de sua máquina hospedeira, que é uma maneira bem conveniente de trabalhar com o mininet. Seguindo este procedimento, o seu ambiente mininet estará pronto para uso. Se preferir, consulte a seção final de referências úteis para tutoriais mais detalhados.

Recomendamos o uso do MobaXTerm., que já possui recursos de redirecionamento X11 habilitado, caso você esteja trabalhando em um ambiente Windows, para acessar os terminais da VM Linux recém-criada.

Configuração e uso do mininet

Com esta seção, você será capaz de utilizar a API do mininet e sua sintaxe de comandos para configurar e executar uma topologia.

Esta etapa pressupõe que você está logado na VM como usuário `mininet`. Um dos requisitos para a execução do mininet é um script python que descreva a topologia da rede virtual. Neste trabalho será utilizado uma topologia predefinida em `~/mininet/custom/topo-2sw-2host.py`, com algumas modificações. Esse arquivo descreve uma topologia com dois hosts ligados a dois switches diferentes, interligados por um link. Vamos inserir parâmetros de qualidade do enlace, como latência, banda e taxa de perda para simular uma rede local entre os links dos hosts criados. Para isso, vocês vão editar as linhas do arquivo que estavam da forma

```
self.addLink(leftHost, leftSwitch)
self.addLink(rightSwitch, rightHost)
```

para

```
self.addLink(leftHost, leftSwitch, bw=10, delay='10ms')
self.addLink(rightSwitch, rightHost, bw=10, delay='10ms')
```

e a linha que estava da forma `self.addLink(leftSwitch, rightSwitch)`
para `self.addLink(leftSwitch, rightSwitch, bw=1, delay='100ms')`

Isso quer dizer que os links dos hosts terão 10 Mbps de banda, e um atraso de 10 ms, enquanto o link entre switches terá uma banda menor (1 Mbps) e um atraso maior (100 ms). Será necessário variar esses parâmetros para os testes de desempenho, como detalhado nas próximas seções. Feito isso, teste sua topologia:

```
$ sudo mn --link tc --custom ~/mininet/custom/topo-2sw-2host.py --topo mytopo
  --test iperf
```

Note que a flag `--test iperf` realiza um teste de desempenho entre os hosts da topologia e retorna do comando `mn`. **Essa flag não será usada para execução de programas teste nos hosts virtuais.**

Trabalhando no ambiente

Todos os programas deverão ser feitos em python, para ficar mais fácil executá-los no mininet. Se os programas forem escritos em outra máquina que não a VM do mininet, basta copiá-los com `scp` para o home do usuário mininet na VM. As etapas para execução de programas no mininet são as seguintes:

1. Logue na VM com `$ ssh -Y mininet@<endereco-host-only-da-vm>`. A flag `-Y` permite redirecionamento X11, necessário para disparar terminais individuais para cada host nas etapas seguintes.
2. Supondo que a topologia foi adequadamente editada, inicie o mininet com

   ```
   $ sudo mn -x --link tc --custom ~/mininet/custom/topo-2sw-2host.py
   ```

   Isso abrirá um terminal por elemento da rede (host, switch ou controlador), desde que a flag `-x` tenha sido incluída. Caso não use -x, dentro do console do Mininet, basta executar o comando `xterm` com o nome do host onde se deseja abrir o terminal.
3. Execute um programa em cada terminal de host, como um script python comum. Você pode executar um servidor em `h1` e um cliente em `h2`, por exemplo. Além disso, **o cliente pode realizar medições de tempo total de transmissão ou vazão e registrar os resultados.**

O problema: transferência de arquivos usando janela deslizante

Na página do curso no moodle você encontra o código para dois arquivos, cliente e servidor, que fazem a transferência de um arquivo pré-definido.

O programa cliente, ao ser disparado, espera pela transferência de um arquivo, fazendo uma abertura ativa de um canal de comunicação. O programa transmissor, que deve ser disparado primeiro, faz a abertura passiva do canal de comunicação, espera a conexão e então envia os dados. Esses programas foram escritos como sendo a camada de aplicação, considerando a interface definida para um protocolo de janela deslizante, o PJD, ainda não implementado.

Sua função é implementar o módulo Python pjd.py, que pode ser usado para fazer o envio de dados por um protocolo de janela deslizante, utilizando por baixo o protocolo UDP.

Você deve implementar, idealmente, um protocolo de janela deslizante sobre UDP (consulte o livro texto para alguns detalhes do código). O tamanho da janela pode ser fixo, mas deve ser ajustável por uma constante no módulo pjd. O protocolo deve idealmente operar no modo de retransmissão seletiva, com a janela do receptor de tamanho igual à do transmissor (definidas em tempo de compilação). Uma implementação que opere no modo *go-back-n* (janela de recepção igual a um) terá uma penalização de 10% da nota final. Um protocolo para-e-espera também é aceitável, mas receberá uma penalização de 25% da nota final.

Essa fonte pode ajudá-lo com detalhes dos modos de operação da janela deslizante.

Idealmente, seu protocolo deve implementar controle de fluxo: se o programa cliente for mais lento que o transmissor, o servidor pode ter que parar de enviar momentaneamente porque a janela do receptor pode ficar cheia de dados ainda não lidos.

Transmissor e receptor, dentro do módulo pjd, deverão lidar com diversos eventos que podem ocorrer sem ordem prédefinida: retransmissões vão exigir um mecanismo de temporização; confirmações podem avançar a janela e permitir o avanço de mais dados; chamadas de send() pela aplicação devem colocar dados na janela ou bloquear até que isso seja possível; chamadas de recv() devem retirar dados da janela de recepção (se possível) retornar ao programa que chama. A forma recomendada para lidar com esses eventos é através do uso de threads Python[2] e com o uso de um timer, possivelmente associado ao socket criado para a conexão.

Simulação de erros na transmissão

O protocolo UDP, usado pela camada de baixo nível, é por natureza um protocolo não confiável. Entretanto, em uma rede local de boa qualidade com máquinas de desempenho compatível, é possível que grande parte da comunicação se dê sem qualquer erro. Para testar o comportamento do protocolo sob condições de erro vamos utilizar o mininet e programar o link entre os dois switches da rede simulada com uma certa taxa de erros. Seu protocolo deve, portanto, utilizar um método de checksum para detectar erros nos pacotes, além de lidar com pacotes perdidos, obviamente.

Saída do programa

Para se observar o desempenho do protocolo e acompanhar a aplicação, diversas estatísticas devem ser coletadas e exibidas, tanto no servidor quanto no cliente. A cada execução, o protocolo PJD deve exibir informações tais como:

- tempo total decorrido desde o início do sistema (ou da última chamada da função de estatísticas);
- total de bytes trafegados e taxa média de envio/recepção;
- total de quadros trafegados e taxa média de envio/recepção;
- total de retransmissões observadas;
- número de confirmações enviadas/recebidas;
- número de quadros duplicados observados;

[2] https://docs.python.org/2/library/threading.html

- quantos erros de cada tipo (pacotes corrompidos/perdidos) foram observados.

Outras estatísticas que sejam apropriadas para o protocolo específico implementado também pode ser incluídas (por exemplo, para um protocolo *go-back-n*, seria interessante determinar quantos pacotes foram recebidos corretamente mas descartados).

Decisões de projeto

Alunos devem trabalhar em duplas; trabalhos individuais, apesar de não serem recomendados, serão aceitos. As notas dos trabalhos serão definidas por meio de entrevistas, onde a qualidade do trabalho da dupla e o domínio de cada aluno sobre o projeto serão considerados nas notas finais.

Tipo de protocolo implementado

Como mencionado anteriormente, um aspecto a definir é a forma de tratamento de erros. Projetos que implementem corretamente um protocolo de janela deslizante receberão notas mais altas em relação àqueles que implementarem apenas um protocolo do tipo para-e-espera. Entretanto, trabalhos que não funcionarem serão penalizados independentemente do protocolo (isto é, melhor um protocolo simples que funciona que um complicado que não funciona).

Detalhes de funcionamento da janela deslizante

Qual o tipo de tratamento de erros adotado, *go-back-n*, repetição seletiva, confirmação seletiva, uso da informação de ACKs duplicados? Seu projeto deve indicar claramente qual a técnica utilizada e qual o impacto dessa escolha sobre a implementação.

Tratamento de temporizações

Como serão implementadas e tratadas as temporizações? Será usado um temporizador para cada pacote na janela, ou um temporizador por janela (há compromissos óbvios de complexidade e eficiência).

Isolamento da camada de enlace

Tanto o cliente quanto o servidor de transferência de arquivos possuem um código simples sequencial: cliente envia dados e servidor os recebe, ambos através de chamadas ao PJD. O protocolo de entrega confiável deve ser implementado de forma a não afetar essa estrutura. Esse é o tipo de requisito com que se deparam os projetistas de protocolos em geral, sejam eles protocolos de kernel ou protocolos de aplicação, como um protocolo de compras em comércio eletrônico. Já que a abstração de camadas define esse isolamento, a implementação deve obedecer a ele.

Além disso, temporizações podem ocorrer a qualquer instante. Não há como escrever um código sequencial que apenas pare esperando por dados de um ponto específico.

Submissão eletrônica

A entrega eletrônica deve ser feita através do Moodle e deve constar de um arquivo do tipo zip ou tar.gz contendo todo o código fonte desenvolvido e um relatório do projeto e implementação.

O código implementado e o relatório final devem ser entregues eletronicamente pelo moodle/minha.ufmg. O relatório deve incluir as informações sobre as questões de implementação discutidas anteriormente, o funcionamento do protocolo, estruturas de dados utilizadas, a solução utilizada para temporizações, etc., além de discutir o impacto final das decisões de projeto, detalhes de implementação e medições de desempenho do protocolo implementado.

O relatório não deve incluir a listagem dos programas. O arquivo de entrega conterá os arquivos do código desenvolvido, mas não deve conter arquivos objeto/compilados

Uma *sugestão* de estrutura para esse relatório:

1. Introdução: descrição do objetivo do trabalho

2. Detalhes do protocolo de entrega confiável implementado.
   Certifique-se de incluir informações como: tipos de mensagens previstas, formato dos pacotes para cada tipo de mensagem, se o protocolo é do tipo pára-e-espera ou janela deslizante, qual a janela do receptor, qual o mecanismo de recuperação de erros, qual a técnica escolhida para a detecção de erros, qual o mecanismo de temporização e retransmissão adotado, etc.

3. Implementação do protocolo: decisões de implementação, como estrutura geral do programa, solução adotada para temporizações, estruturas de dados utilizadas, técnica utilizada para lidar com eventos assíncronos, etc.

4. Dados de desempenho com e sem a inserção de erros na transmissão. Parâmetros que podem ser interessantes observar são o impacto da taxa de erros sobre o desempenho final e o impacto diferentes bandas/atrasos no link entre os switches. Se forem gerados gráficos, esses devem usar escalas adequadas (logarítmicas ou lineares, conforme o caso), os eixos devem ser identificados e possuir claramente a identificação das unidades em cada um. Os resultados devem ser apresentados por linhas retas interligando os pontos medidos, que devem ser destacados com marcas claras. Se um gráfico contiver mais que um conjunto de pontos as linhas devem ser claramente identificadas. Gráficos gerados pelo gnuplot normalmente têm essas características como padrão.

5. Análise: discuta os resultados observados. Os resultados foram de acordo com o esperado? Você é capaz de explicar por que o protocolo se comporta como observado? Houve algum elemento claramente de destaque nos resultados que merece uma análise especial (por exemplo, um problema de implementação que não havia sido previsto inicialmente)?

6. Conclusão: como todo trabalho técnico, algumas palavras finais sobre o resultado do trabalho, tanto das observações quanto do seu aprendizado sobre o assunto. (Não vale xingar o professor!)

Considerações finais

Este documento descreve em linhas gerais o trabalho e oferece alguns detalhes sobre como projetá-lo e implementá-lo. O trabalho lhes dá liberdade no projeto do protocolo e na sua implementação. Nem todos os detalhes estão claramente apresentados e definidos. Parte do trabalho é avaliar como os grupos abordam uma especificação parcialmente aberta e a transformam em um produto acabado. O domínio das técnicas envolvidas é parte do trabalho e não é considerado pré-requisito.

Observações Gerais

1. **Programas cuja saída não sigam o comportamento definido, ou que exijam alterações nas partes pré-definidas do código, ou que sejam submetidos de forma incorreta (especialmente no que diz respeito à submissão eletrônica) serão penalizados no processo de avaliação.**

2. **Dúvidas:** usem o moodle (minha.ufmg). Se o problema envolver trecho de código, envie mensagem para o monitor (Luis Felipe luisf@dcc.ufmg.br) com cópia para o professor. Tentaremos responder toda pergunta em menos de 48 horas.
   Respostas a perguntas que sejam consideradas de interesse geral serão colocadas no moodle (a privacidade de inocentes será preservada).

3. **Comece a fazer este trabalho logo, enquanto o problema está fresco na memória e o prazo para terminá-lo está tão longe quanto jamais poderá estar.**

4. **Implemente o trabalho por partes:** Por exemplo, crie o formato da mensagem e tenha certeza que o envio e recebimento da mesma está correto (antes que se envolver com a lógica da entrega confiável e entrada/saída). Depois, veja se a progressão de números de sequência está acontecendo como esperado nas duas partes. Conforme também se o mecanismo de retransmissão e avanço da janela está ocorrendo na ordem correta.

5. **Atenção:** por se tratar de um trabalho complexo, dê uma atenção especial para a clareza do código e procure mantê-lo o mais simples possível. Lembre-se que você poderá ter que explicar sua solução durante uma entrevista, incluindo as decisões de implementação.

6. **Você pode discutir soluções com os colegas, mas não é permitido o compartilhamento de trechos de código.**

7. **Vão valer pontos clareza, indentação e comentários no programa.**

Referências úteis

- Configuração de uma rede *host-only* no VirtualBox
- Programação com sockets: python 2; python 3
- Tutorial detalhado das funcionalidades do Mininet
- Computing the Internet Checksum (RFC)
- Módulo threading threading; Retransmissões

Última alteração: 7 de maio de 2016