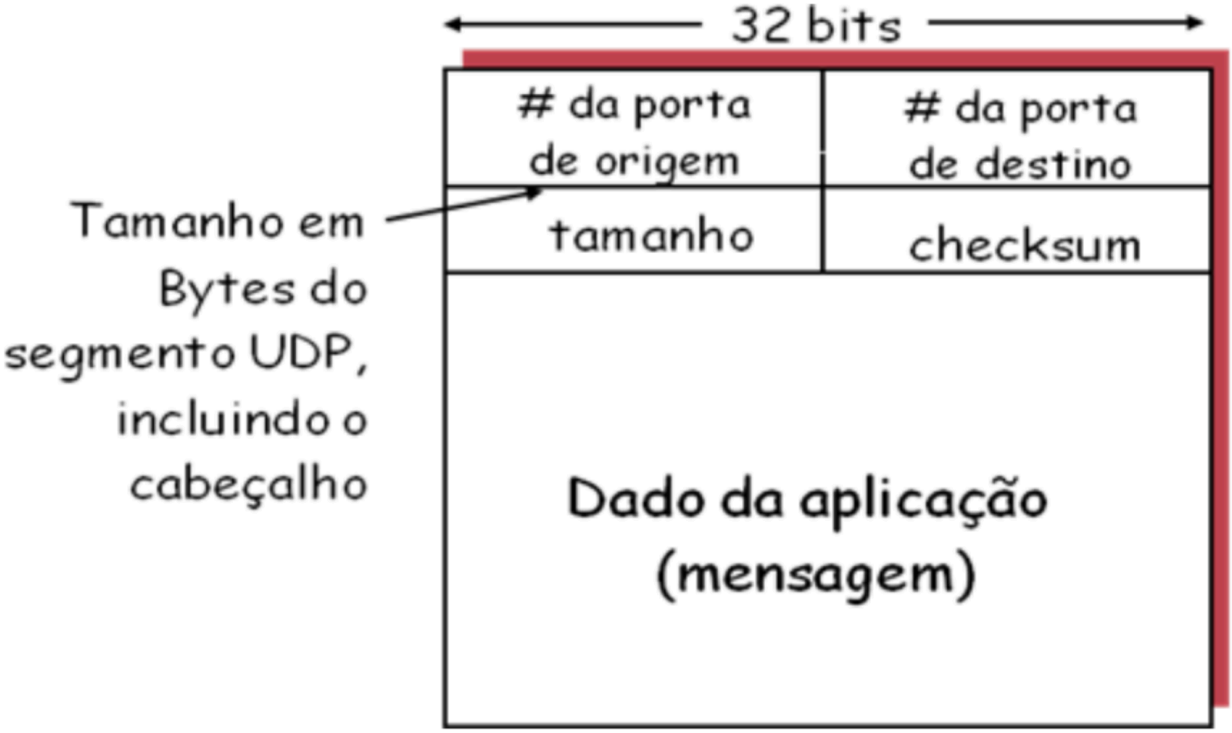
Lab 2 : Protocolos de Transporte – UDP/TCP

Este laboratório discute a forma de operação dos dois protocolos mais comuns na camada de transporte: UDP (User Datagrama Protocol) e TCP (Transmission Control Protocol). Os exercícios deste laboratório são baseados no livro "Mastering Networks – An Internet Lab Manual – Jörg Liebeherr, Magda El Zarki – Addison Wesley – 2004 – ISBN 0-201-78134-4.

Introdução

UDP é um protocolo simples, não orientado à conexão, cujo objetivo é a troca de mensagens entre duas aplicações (uma emissora e uma receptora).

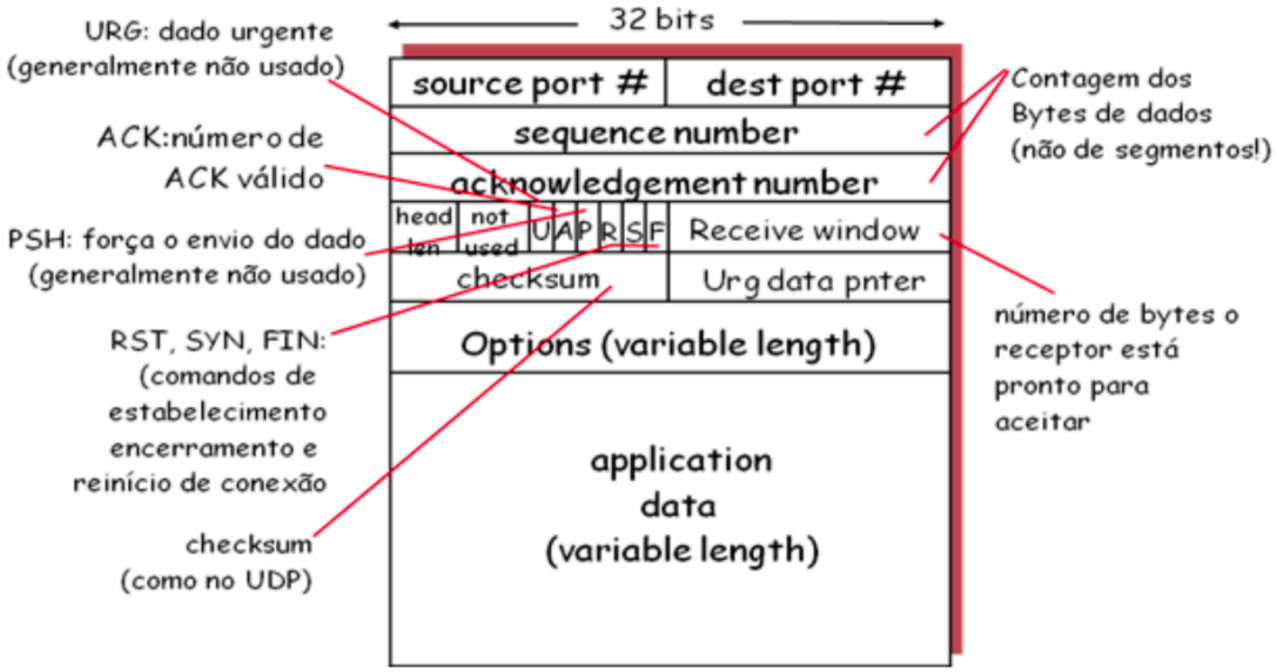
O protocolo UDP consiste em um datagrama UDP, ilustrado na Figura abaixo, composto por um pequeno cabeçalho adicionado ao conteúdo recebido da camada superior (camada de aplicação) . Quando transmitido, o datagrama UDP é encapsulado em um cabeçalho IP e enviado ao destino. Para cada conteúdo (mensagem) enviado, existe um único datagrama UDP associado.



O TCP, por sua vez, é um protocolo de operação mais complexa e, diferentemente do UDP, o TCP é orientado à conexão. Isto é, um cliente TCP primeiramente estabelece uma conexão com um servidor TCP antes do início da

transmissão efetiva dos dados. Após o estabelecimento da conexão, a transferência dos dados ocorre em ambos os sentidos: do cliente ao servidor e vice-versa.

A unidade de dados do TCP é denominada de segmento_TCP, ilustrado na Figura abaixo, o qual é formado por um cabeçalho TCP e uma carga que contém os dados da aplicação. A aplicação emissora submete o dado ao TCP como um fluxo contínuo de bytes (fluxo de dados) sem indicação de limites para tal fluxo. O lado TCP do emissor decide quantos bytes serão incluídos no segmento a ser transmitido.



Uma característica básica do TCP é a garantia da entrega dos segmentos transmitidos. Para tal, o protocolo utiliza (1) checksum para deteção de erro, (2) numeração da sequência de bytes para indicação da posição do byte no fluxo de dados e (3) reconhecimento e temporização para detecção de segmentos perdidos ou corrompidos. Além disso, o lado receptor do TCP envia um segmento de controle (ACK) ao emissor para notificar o recebimento de um segmento de dados. Vários segmentos TCP pertencentes a um mesmo fluxo podem ser reconhecidos através de um único ACK. Quando o lado emissor do TCP não recebe um ACK dentro de um intervalo predeterminado, é assumido que o segmento foi perdido e o mesmo é retransmitido pelo emissor.

O TCP possui dois mecanismos que controlam a quantidade de dados que o emissor pode enviar. Primeiramente, o lado receptor do TCP informa ao lado emissor quantos bytes o emissor pode transmitir sem recebimento de

reconhecimento (ACK). Esse mecanismo denomina-se controle de fluxo (flow control). Segundo, quando a rede encontra-se sobrecarregada e segmentos TCP começam a ser perdidos devido ao congestionamento, o lado emissor do TCP reduz a taxa de transmissão do tráfego. Esse mecanismo é denominado de controle de congestionamento (congestion control).

Configuração

> Atenção: Caso algum erro seja gerado, tente executar os comandos como super usuário (sudo <comando>)

A atividade será baseada na topologia indicada na Figura 3. Primeiro, vamos começar com os passos de configuração.

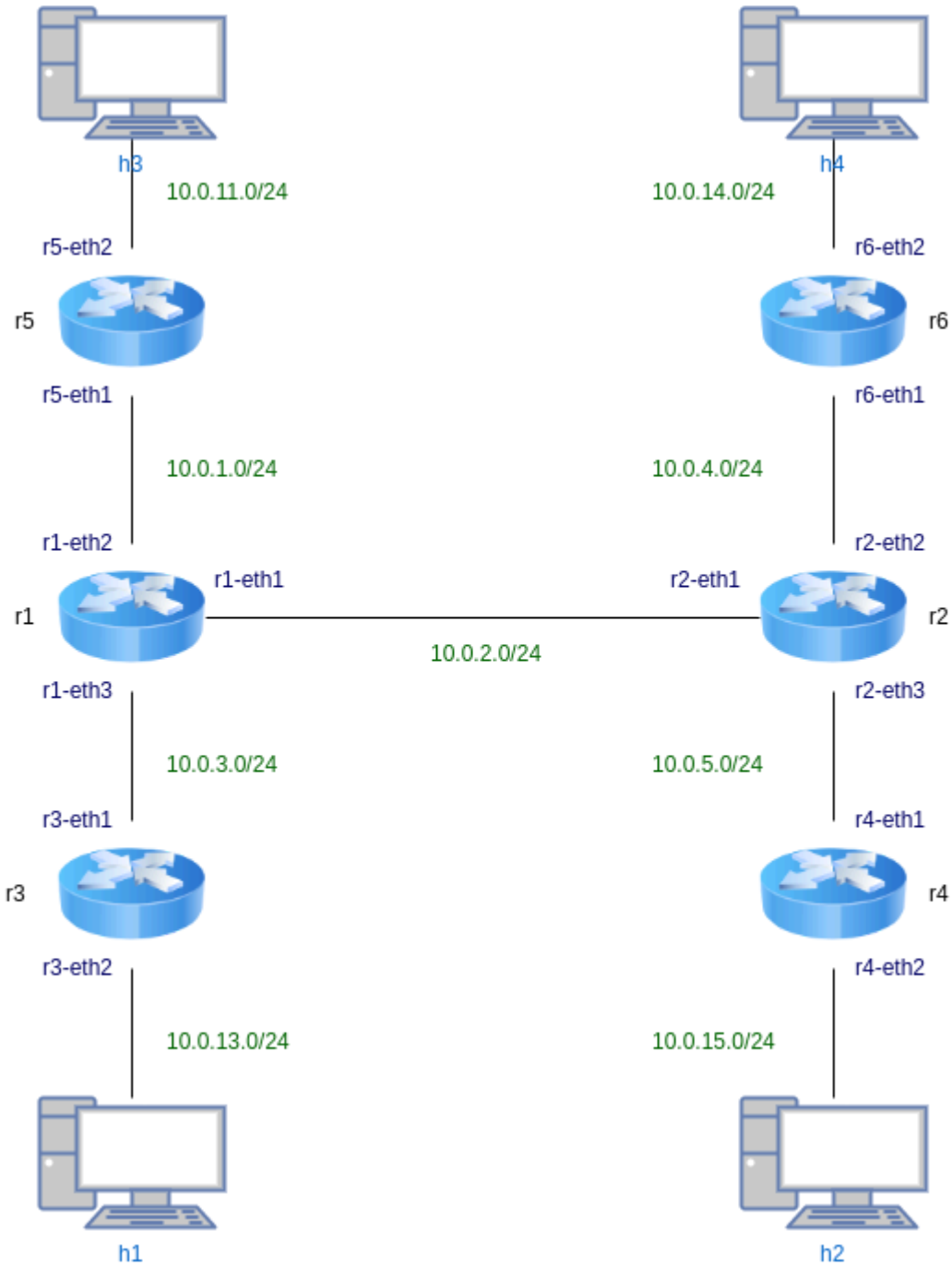
Para o laboratório é necessário a instalação do pacote D-ITG (versão 2.8.1) que implementa um gerador de tráfego. O pacote D-ITG pode ser instalado utilizando os seguintes comandos:

```
cd /home/
wget https://github.com/intrig-unicamp/ea080/raw/master/software/D-ITG-2.8.1-r10
unzip D-ITG-2.8.1-r1023-src.zip
cd D-ITG-2.8.1-r1023/src/
make multiport=off bursty=off
```

O script com a topologia indicada na figura (arquivo de configuração tcp.py (http://tcp.py)) está disponível em https://github.com/intrig-unicamp/ea080.git (https://github.com/intrig-unicamp/ea080.git)

```
cd /home/wifi/
git clone https://github.com/intrig-unicamp/EA080-2S2021.git
cd EA080-2S2021
cd lab2
```

Os exercícios deste roteiro têm como objetivo testar e verificar o funcionamento dos protocolos UDP/TCP. Tais exercícios serão realizados com base na topologia exemplificada na Figura abaixo, contida no arquivo mininet chamado tcp.py (http://tcp.py).



Para dar início aos exercícios, execute o arquivo tcp.py (http://tcp.py) (encontrado em /EA0802S2021/lab2/).

```
sudo python tcp.py
```

Exercícios

Exercício 1

A partir da topologia executada confira se os nós e enlaces conferem com os mostrados na Fig. 3.

```
mininet> nodes
mininet> links
```

Ou para uma melhor visualização completa da topologia:

```
mininet> net
```

Abra um terminal externo para o host h1.

```
mininet> xterm h1
```

Crie um diretório com nome h1:

```
mkdir h1
cd h1
```

Neste Laboratório vamos utilizar a ferramenta chamada netcat, cuja função é estabelecer conexões entre hosts, ou seja, entre as máquinas interligadas por uma rede, sendo esta rede interna ou externa.

Basicamente o que o netcat faz é estabelecer uma conexão entre dois computadores, utilizando os protocolos TCP ou UDP como meio de tráfego para que seja possível escrever dados em conexões de rede. Examine as opções através do comando abaixo.

```
man netcat
```

Execute o programa Wireshark em background no host h1

```
wireshark &
```

inicie uma captura na interface h1-eth1 e instancie um receptor netcat para receber tráfego UDP usando o seguinte comando:

```
nc -l 4444 -u > data1.txt
```

Voltando à tela principal do mininet, abra um terminal externo para o host h4.

```
xterm h4
```

Crie um diretório com nome h4:

```
mkdir h4
cd h4
```

Neste host vamos criar um arquivo de tamanho 1200 bytes para ser enviado para o host h1:

```
truncate -s 1200 data1.txt
```

Agora vamos enviar o arquivo para h1 com o comando de netcat. Nesse caso, estamos enviado o arquivo para o ip 10.0.13.3, utilizando UDP e com um timeout de 3 segundos.

```
h4> nc -w 3 10.0.13.3 4444 -u < data1.txt
```

Observe o tráfego capturado através do Wireshark no host h1

> P: Quantos pacotes foram transmitidos para cada datagrama UDP?
> P: Compare o total de bytes transmitidos incluindo os cabeçalhos Ethernet, IP e UDP
> relativamente ao total de dados de aplicação (data) transmitidos.

Repita o processo de envio do pacote [1]:

```
h1> nc -l 4444 -u > data1.txt
h4> nc -w 3 10.0.13.3 4444 -u < data1.txt
```

> P: Inspecione os campos dos cabeçalhos UDP. Quais campos nos cabeçalhos não foram alterados nos diferentes pacotes transmitidos? (compare com o primeiro envio)

[1] Use ctrl+c para finalizar o processo anterior em h1

Exercício 2

Esse exercício repete o exercício 1 tendo como alvo o protocolo TCP. Execute o Wireshark e instancie um receptor netcat para receber tráfego TCP no host h1 usando o seguinte comando:

```
nc -l 4444 > data1.txt
```

Agora vamos enviar o arquivo do host h4 ao h1 com o comando de netcat (executado no terminal do host 4):

```
nc -w 3 10.0.13.3 4444 < data1.txt
```

> P: Quantos pacotes TCP foram trocados durante a comunicação?
> P: Quais são os tamanhos dos segmentos TCP?
> P: Calcule o total de bytes transmitidos em ambas as direções (todos os pacotes TCP enviados por h1 e também por h2), incluindo os cabeçalhos Ethernet, IP, TCP e os dados transmitidos pela aplicação (payload). Qual proporção desse valor foi destinado apenas aos dados da aplicação?

Exercício 3

No host h4 vamos criar um arquivo de tamanho 4000 bytes para ser enviado para o host h1.

```
truncate -s 4000 data2.txt
```

Repita o envio de dados UDP e TCP dos exercícios 1 e 2 utilizando pacotes com 4000 Bytes de tamanho (data2.txt ao invés de data1.txt).

> P: Compare os resultados obtidos para o envio de 1200 bytes e 4000 bytes tanto para TCP quanto para UDP, procurando identificar as principais diferenças entre os dois protocolos.
> P: Quantos pacotes UDP/TCP foram trocados durante a transferência dos dados?
> P: Quais são os tamanhos de dados de aplicação transmitidos nos pacotes UDP/TCP?

Exercício 4

Sendo o TCP um protocolo orientado a conexão, uma conexão apenas é iniciada quando um cliente TCP envia uma requisição de conexão a um servidor TCP. O servidor deve estar em funcionamento quando o pedido de conexão é enviado.

O TCP requer três pacotes para abertura de conexão. Esse procedimento é denominado three-way handshake. Durante o processo de handshaking, o cliente TCP e o servidor TCP negociam parâmetros essenciais à conexão TCP, incluindo o (1) número de sequência inicial, o (2) tamanho máximo do segmento (MSS) e o (3) tamanho da janela para o controle de fluxo baseado em janela deslizante.

Por outro lado, o TCP requer 3 ou 4 pacotes para fechar a conexão. Cada lado da conexão é fechado separadamente. O TCP utiliza bits como flags no cabeçalho do protocolo indicando que o protocolo carrega informações de controle. As flags envolvidas na abertura/encerramento de conexão são SYN, ACK e FIN.

Utilize o comando netcat para estabelecimento de uma conexão TCP entre o h4 e o h1. Observe os pacotes de controle utilizados na abertura e encerramento da conexão TCP.

Execute o Wireshark e então instancie um receptor netcat para receber tráfego TCP usando o seguinte comando (em h1):

```
h1> nc -l 4444
```

No host h4 execute o comando:

```
nc 10.0.13.3 4444
```

> P: Identifique os pacotes utilizados no three-way handshake. Informe quais flags são marcadas no cabeçalho TCP e explique como estas informações são interpretadas pelo servidor e o cliente TCP;
> P: Quais são os Números de Sequência (sequence number) trocados nessa comunicação inicial?

Ainda na tela do netcat (h4) envie um texto ao servidor (h1) e verifique se o mesmo foi recebido pelo cliente (h1):

```
h4> Hello
```

> P: Identifique o primeiro pacote contendo dado. Qual é o número de sequência do primeiro byte do dado da aplicação enviado pelo cliente ao servidor TCP?

O cliente e o servidor informam o tamanho da janela definindo a quantidade máxima de dados que podem ser enviados em cada direção sem reconhecimento e negociam também a quantidade máxima de bytes em um segmento TCP.

> P: Determine os valores dos tamanhos de janela indicados pelo cliente e pelo servidor;
>
> P: Qual é o valor do MSS (Maximum Segment Size) negociado entre o cliente e o servidor?
>
> P: Ilustre os números de sequência e de reconhecimento utilizados.
>
> P: Feche a conexão no h4 (ctrl+c). Identifique quais são os pacotes envolvidos no fechamento da conexão. Quais flags estão marcadas no pacote? Explique como estas flags são interpretadas pelo servidor e pelo cliente TCP.

Exercício 5

Para as próximas atividades utilizaremos os Hosts h2 e h3 para geração de tráfego de fundo ao tráfego entre o Host h1 e o Host h4.

Primeiro vamos preparar os dados para o teste no host h4:

```
h4> truncate -s 100M data3.txt
```

Utilize o gerador de tráfego D-ITG nos hosts virtuais para gerar tráfego TCP ou UDP. No caso, é necessário definir uma máquina como origem do tráfego (ex. h3) e outra máquina como destino do tráfego (ex. h2). O emissor ITG (ITGSend) transmite os pacotes segundo os parâmetros definidos para geração do tráfego (pacotes/seg, tamanho de pacotes, etc.).

O comando a seguir exemplifica o envio de pacotes UDP:

```
>ITGSend –a 10.0.0.y –C x –c y –t z –T UDP -l udp.log -x udp.log
```

- -a:host destino (10.0.0.y);
- -C:taxa de pacotes (x = pacotes/seg);
- -c:tamanho do dado (y = Bytes, default = 512 B);
- -t:duração do envio de pacotes (z = milisegundos);
- -T:protocolo utilizado (UDP|TCP|ICMP);
- -l:nome do arquivo de log gerado no emissor e -x nome do arquivo de log gerado no receptor.

Você pode examinar mais opções através do comando >ITGSend –help
Abra um terminal externo para o host h2 e instancie um receptor ITGRecv para receber tráfego usando os seguintes comandos:

```
mininet> xterm h2
h2> cd /home/wifi/D-ITG-2.8.1-r1023/bin/
h2> ./ITGRecv
```

No Host h3 instancie um emissor ITGSend para gerar tráfego de fundo UDP através dos comandos:

```
mininet> xterm h3
h3> cd /home/wifi/D-ITG-2.8.1-r1023/bin/
h3> ./ITGSend -a 10.0.15.3 -C 1000000000000 -c 40000 -t 60000 -T UDP
```

Execute o Wireshark no host h1 e instancie um receptor netcat para receber tráfego TCP usando o seguinte comando:

```
h1> nc -l -p 4444
```

No host h4 execute o comando:

```
h4> for i in {1..10} ; do cat data3.txt ; done | nc 10.0.13.3 4444
```

Deixe rodar o comando por 30 segundos e depois pare a captura no Wireshark (utilize a coluna Time como referência).

Discuta o comportamento da conexão TCP entre Host h4 e Host h1 através do gráfico Time-Sequence Graph (tcptrace[2]). O tcptrace é um gráfico do número de sequência versus tempo que também inclui os valores ACK e o tamanho da

janela enviados pelo outro extremo da conexão. Utilize o zoom no gráfico (tecla i aumenta o zoom enquanto a tecla o reduz o zoom) e selecione um trecho da transmissão.

> P: Faça uma análise das mensagens trocadas nesse caso
> P: Faça uma análise do comportamento da conexão TCP observado no experimento através da análise dos gráficos Throughput e Round-Trip Time.

[2]No Wireshark em Statistics>TCP StreamGraph>Time-Sequence Graph(tcptrace).

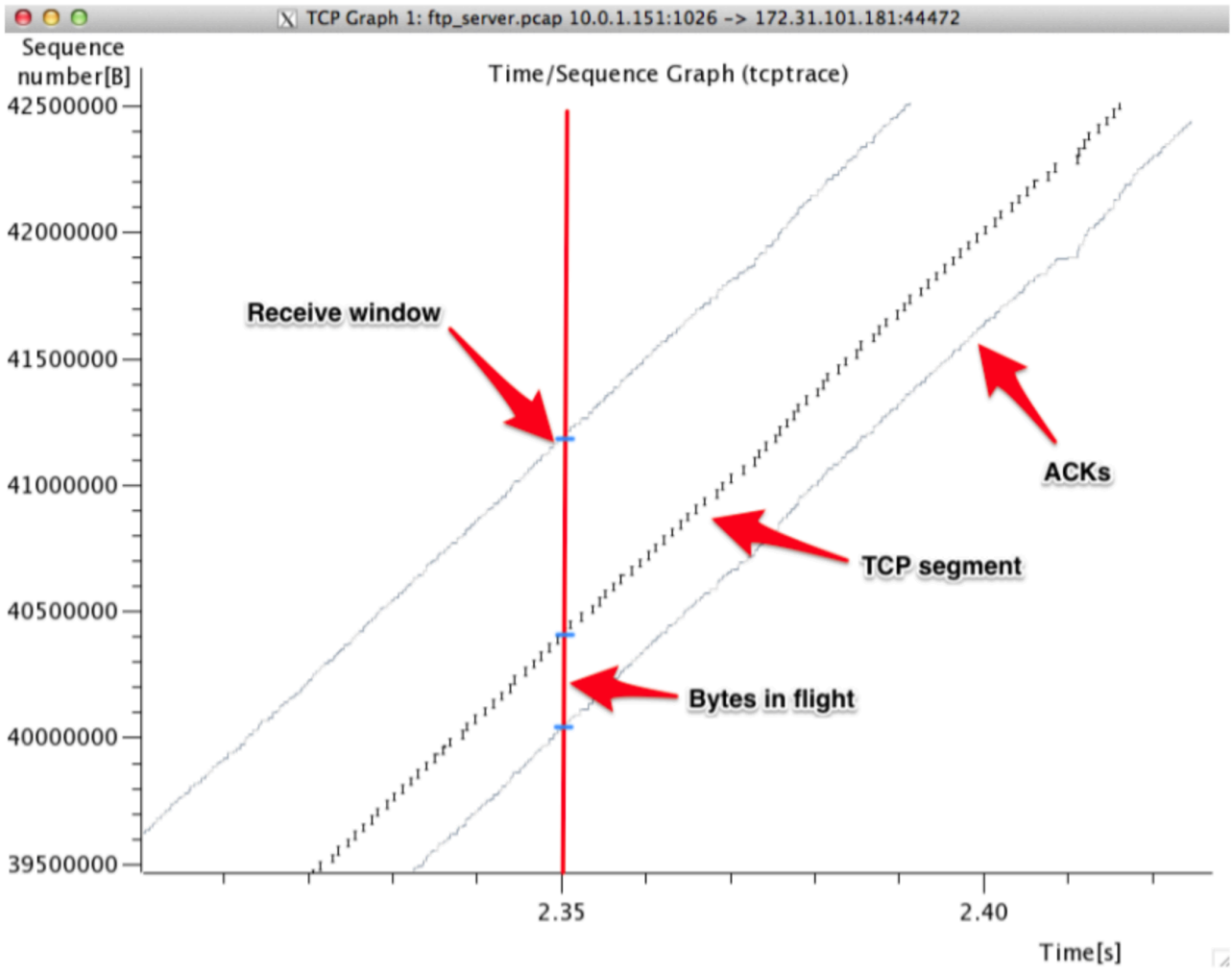
Apêndice: Entendendo o gráfico de sequência temporal no Wireshark

Introdução

Quando estou solucionando um problema de desempenho, uma das primeiras ferramentas que utilizo no wireshark é o Time-Sequence Graph (tcptrace) ou gráfico de sequência temporal, que pode ser encontrada em Statistics > TCP StreamGraph > Time-Sequence Graph (tcptrace).

O gráfico da sequência temporal mostra um fluxo de dados ao longo do tempo. Por definição, um fluxo de dados é unidirecional. Portanto, se um cliente está baixando um arquivo de um servidor FTP, você deve clicar em um pacote que sai do servidor antes de gerar o gráfico. Lembre-se, o gráfico mostrará apenas os dados que fluem em uma direção.

A Figura abaixo mostra uma captura de tela ampliada com algumas observações.



O eixo x é o tempo, mostrado em segundos. O eixo y é o número de sequência TCP. Os números de sequência são representativos dos bytes enviados. O número de sequência aumenta em 1 para cada 1 byte de dados TCP enviados. O ideal é ver uma linha suave subindo para a direita. A inclinação da linha é a largura de banda teórica da conexão. Quanto mais inclinada a linha, maior será a taxa de transferência.

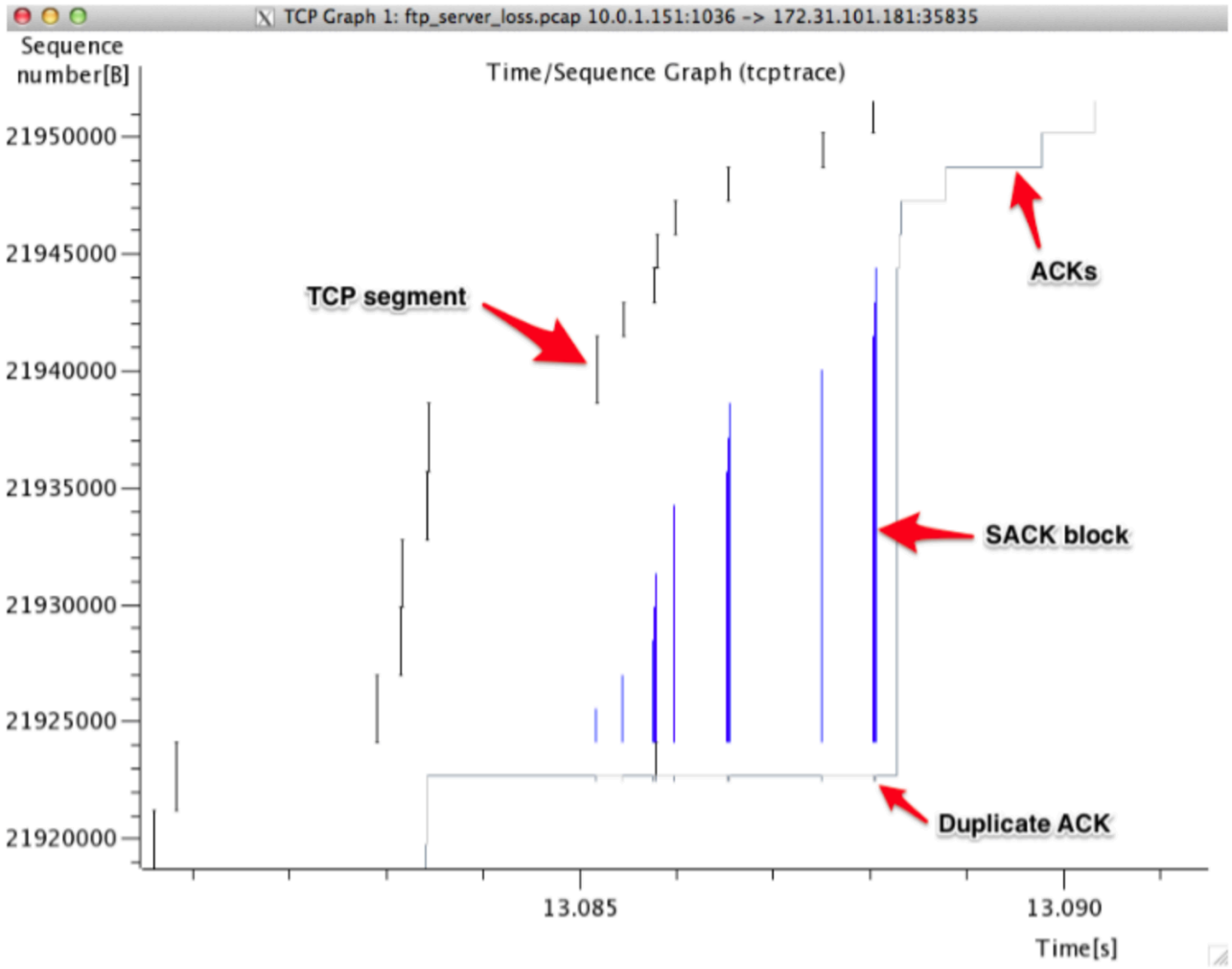
As pequenas linhas verticais em formato I pretas representam segmentos de dados TCP. Quanto mais longo for o I, mais dados o segmento possui. A linha cinza abaixo são os ACKs do receptor. A distância entre os ACKs e os dados TCP em um determinado ponto no tempo representa os bytes em trânsito enviados pelo receptor cujo recebimento ainda não foi acusado pelo receptor. Assim, se em 2,35 segundos o servidor estiver enviando 40.400.000 bytes e receber o ACK de 40.000.000, haverá 400.000 bytes não reconhecidos em trânsito. (As linhas vermelha e as marcas azuis em 2,35; não faz parte do gráfico).

A linha superior representa a janela de recepção (receive window) calculada do cliente, obtida através da soma do número ACK e da janela de recepção atual anunciada. Caso o ACK atual seja 40.000.000 e a janela de recepção anunciada

seja de 1.200.000, então a janela de recepção calculada será 41.200.000.

A distância entre o número de sequência TCP atual (40.400.000) e a janela de recebimento calculada (41.200.000) é a quantidade de dados que o cliente ainda pode reter em seu buffer (800.000).

A Figura abaixo mostra algumas informações adicionais:



Ainda temos os dados do segmento TCP e os ACKs representados como antes. Mas agora temos duas coisas novas em relação à perda e recuperação de dados. As ACKS duplicadas são representadas como pequenas marcas (linhas) na parte inferior da linha ACK. Os blocos SACK são as linhas azuis acima das marcas, ou seja, ACKS duplicados.

Algumas dicas de utilização da ferramenta:

- Você pode usar a tecla 'i' para aumentar o zoom na posição atual do mouse
- Você pode usar a tecla 'o' para diminuir o zoom na posição atual do mouse
- Você pode segurar e arrastar o gráfico com o botão direito do mouse
- Você pode segurar com um clique esquerdo e arrastar um retângulo para

ampliar uma região

- Você pode clicar com um único clique esquerdo em um segmento ou ACK para ir a esse pacote no pcap (muito útil)