Escola Politécnica da Universidade de São Paulo
Departamento de Engenharia de Telecomunicações e Controle

PTC3450 - Redes de Comunicação
1o semestre 2017

Lista de Exercícios Suplementares 2

1) [Kurose and Ross, 2017, p. 200] Em quais circunstâncias o *download* de um arquivo via P2P é bem mais rápida do que por meio de uma abordagem cliente-servidor centralizada? Justifique sua resposta usando as equações vistas em aula.

2) [Kurose and Ross, 2017, p. 200] Assuma que um *tracker* BitTorrent torna-se repentinamente indisponível. Quais são as consequências desse fato? Arquivos podem continuar sendo baixados?

3) [Kurose and Ross, 2017, p. 207] Considere a distribuição de um arquivo de $F = 15$ Gbits para N pares. O servidor tem uma taxa de *upload* de $u_S = 30$ Mbps e cada par tem uma taxa de *download* de $d_i = 2$ Mbps e uma taxa de *upload* u. Para $N = 10$, 100 e 1 000 e $u = 300$ kbps, 700 kbps e 2 Mbps, prepare uma tabela fornecendo o mínimo tempo de distribuição para cada uma das combinações de N e u tanto para distribuição cliente-servidor como para distribuição P2P.

4) [Kurose and Ross, 2017, p. 208] Considere um sistema DASH para o qual existam N versões de vídeos (em N diferentes taxas e qualidades) e N versões de áudio (em N diferentes taxas e qualidades). Suponha que desejamos permitir que o aplicativo *player* escolha a qualquer instante uma das N versões de vídeo e uma das N versões de áudio.

   a) Se criarmos arquivos de forma que o áudio fique misturado com o vídeo, de modo que o servidor envia apenas um fluxo de mídia em um dado instante, quantos arquivos diferentes o servidor necessita armazenar (cada um com uma URL diferente)?

   b) Se ao invés disso, o servidor armazena separadamente fluxos de áudio e vídeo e faz o cliente sincronizar os fluxos, quantos arquivos diferentes o servidor precisa armazenar?

5) [Kurose and Ross, 2017, p. 200] Os CDNs tipicamente adotam uma de duas filosofias diferentes de alocação de servidores. Nomeie-as e as descreva brevemente.

6) [Kurose and Ross, 2017, p. 208] Qual a estratégia de alocação de servidores adotada pela Netflix para o seu CDN? Como o conteúdo é replicado entre diferentes servidores?

7) [Kurose and Ross, 2017, p. 208] Instale e compile os programas Python `TCPClient` e `UDPClient` em um *host* e o `TCPServer` e `UDPServer` em outro *host*.

   a) Suponha que você rode o `TCPClient` antes de rodar o `TCPServer`. O que acontece? Por que?

b) Suponha que você rode o `UDPClient` antes de rodar o `UDPServer`. O que acontece? Por que?

c) O que acontece se você usar números de porta diferentes nos lados cliente e servidor?

8) [Kurose and Ross, 2017, p. 200] O servidor UDP descrito em aula necessita de apenas um *socket*, enquanto o servidor TCP precisa de 2. Por que? Se o servidor TCP precisa atender n conexões simultâneas, cada uma de um *host* cliente diferente, quantos *sockets* o servidor TCP precisa?

9) [Kurose and Ross, 2017, p. 201] Para a aplicação cliente-servidor TCP descrita em aula, por que o programa servidor precisa ser executado antes do programa cliente? Para a aplicação cliente-servidor sobre UDP, por que o programa cliente pode ser executado antes do programa servidor?

10) [Kurose and Ross, 2013] O UDP e o TCP usam complemento de 1 para suas somas de verificação. Suponha que você tenha as seguintes três palavras de 8 bits: 01010011, 01100110 e 01110100. Qual é o complemento de 1 para as somas dessas palavras? (Note que, embora o UDP e o TCP usem palavras de 16 bits no cálculo da soma de verificação, nesse problema solicitamos que você considere somas de 8 bits.) Mostre todo o trabalho. Por que o UDP toma o complemento de 1 da soma, isto é, por que não toma apenas a soma? Com o esquema de complemento de 1, como o destinatário detecta erros? É possível que um erro de 1 bit passe despercebido? E um erro de 2 bits?

11) [Kurose and Ross, 2013]

a) Suponha que você tenha os seguintes bytes: 01011100 e 01100101. Qual é o complemento de 1 da soma desses 2 bytes?

b) Suponha que você tenha os seguintes bytes: 11011010 e 01100101. Qual é o complemento de 1 da soma desses 2 bytes?

c) Para os bytes do item (a), dê um exemplo em que um bit é invertido em cada um dos 2 bytes e, mesmo assim, o complemento de um não muda.

12) (2014-P1) [Kurose and Ross, 2013] Considere a transferência de um arquivo enorme de L bytes do *host* A para o *host* B. Suponha um MSS de 536 bytes.

(a) Qual é o máximo valor de L tal que não sejam esgotados os números sequenciais do TCP? Lembre-se que o campo de número sequencial no TCP tem 4 bytes.

(b) Para o L que obtiver em (a), descubra quanto tempo demora para transmitir o arquivo. Admita que um total de 66 bytes de cabeçalho de transporte, de rede e de enlace de dados seja adicionado antes que o pacote seja enviado por um enlace de 155 Mbits/s. Ignore controle de fluxo e controle de congestionamento de modo que A possa enviar os segmentos um atrás do outro e continuamente.

Resp: (a) 2^{32} bytes; (b) 248,97 s

13) [Kurose and Ross, 2013] No protocolo genérico SR que estudamos em aula, o remetente transmite uma mensagem assim que ela está disponível (se ela estiver na janela), sem esperar por um reconhecimento. Suponha, agora, que queiramos um protocolo SR que envia duas mensagens de cada vez. Isto é, o remetente enviará um par de mensagens, e o par de mensagens subsequentes somente deverá ser enviado quando o remetente souber que ambas as mensagens do primeiro par foram recebidas corretamente.

Suponha que o canal possa perder mensagens, mas que não as corromperá nem as reordenará. Elabore um protocolo de controle de erro para a transferência confiável unidirecional de mensagens. Dê uma descrição FSM do remetente e do destinatário. Descreva o formato dos pacotes enviados entre o remetente e o destinatário e vice-versa. Se você usar quaisquer procedimentos de chamada que não sejam os vistos em aula (por exemplo, `udt_send()`, `start_timer()`, `rdt_rcv()`, etc.) esclareça as ações desses procedimentos. Dê um exemplo (um diagrama de mensagens para o remetente e destinatário) mostrando como seu protocolo se recupera de uma perda de pacote.

14) [Kurose and Ross, 2013] Considere o protocolo GBN com uma largura de janela remetente de 4 e uma faixa de números sequenciais de 1 024. Suponha que, no instante t, o pacote seguinte na ordem pelo qual o destinatário está esperando, tenha um número sequencial k. Admita que o meio não reordene as mensagens. Responda às seguintes perguntas:

    a) Quais são os possíveis conjuntos de números sequenciais, dentro da janela do remetente no instante t? Justifique sua resposta.

    b) Quais são todos os possíveis valores do campo ACK em todas as mensagens que estão atualmente se propagando de volta ao remetente no instante t? Justifique sua resposta.

15) [Kurose and Ross, 2013] Os *hosts* A e B estão se comunicando por meio de uma conexão TCP. O *host* B já recebeu de A todos os bytes até o byte 126. Suponha que A envie, então, dois segmentos para B sucessivamente. O primeiro e o segundo segmentos contêm 80 e 40 bytes de dados. No primeiro segmento, o número sequencial é 127, o número de porta de partida é 302, e o número de porta de destino é 80. O *host* B envia um ACK ao receber um segmento do *host* A.

    a) No segundo segmento enviado do *host* A para B , quais são os números sequenciais, da porta de origem e da porta de destino?

    b) Se o primeiro segmento chegar antes do segundo, no ACK do primeiro segmento que chegar, qual é o número do ACK, da porta de origem e da porta de destino?

    c) Se o segundo segmento chegar antes do primeiro, no reconhecimento do primeiro segmento que chegar, qual é o número do ACK?

    d) Suponha que dois segmentos enviados por A cheguem em ordem a B. O primeiro ACK é perdido e o segundo chega após o primeiro intervalo de *timeout*. Elabore um diagrama de temporização, mostrando esses segmentos, todos os outros e os ACKs enviados. (Suponha que não haja qualquer perda de pacote adicional.) Para cada segmento de seu diagrama, apresente o número sequencial e o número de bytes de dados; para cada ACK adicionado por você, informe o número do ACK.

Referências

Kurose, J. and Ross, K. (2013). *Redes de Computadores e a Internet: Uma Abordagem Top Down*. Pearson.

Kurose, J. and Ross, K. (2017). *Computer Networking: A Top-Down Approach*. PEARSON EDUC.