Lista de Exercícios 7 - Redes de Computadores
Júlio Melo Campos - 22250349

Seção 3.1 a 3.3

R3 - Considere uma conexão TCP entre o Host A e o Host B. Suponha que os segmentos TCP que viajam do Host A para o Host B tenham o número da porta de origem x e o número da porta de destino y. Quais são os números das portas de origem e destino para os segmentos que viajam do Host B para o Host A?

R: Para os segmentos TCP que viajam do Host B para o Host A, o número da porta de origem e o número da porta de destino serão invertidos em relação aos segmentos que viajam do Host A para o Host B. Ou seja:

- A porta de origem será a porta y (porta de destino dos segmentos A para B);
- Enquanto a porta de destino será a porta x (porta de origem dos segmentos A para B).

R4 - Descreva por que um desenvolvedor de aplicação pode optar por executar uma aplicação sobre UDP em vez de TCP.

R: Um desenvolvedor pode optar UDP pelo TCP, pois há **baixa latência**, já que o UDP é um protocolo sem conexão e não realiza controle de fluxo e verificação de erros como TCP; **eficiência**, pois o UDP não realiza processo de estabelecimento e encerramento de conexão, sem necessidade de enviar confirmações por causa de recebimentos de pacotes, assim reduzindo o overhead, economizando largura de banda e processamento; além do **controle de fluxo de dados**, pois o UDP permite maior controle de envio e gerenciamento de pacotes para o desenvolvedor pelo fato de não existir retransmissão automática em caso de erros; e por fim, **escalabilidade**, onde O UDP se torna mais escalável, pois não exige conexão individual como o TCP e sim, simultânea a todos.

R5 - Por que o tráfego de voz e vídeo é frequentemente enviado por TCP em vez de UDP na Internet atual?

R: É preferível na internet atual o TCP ao invés do UDP para o tráfego de voz e vídeo, mesmo que o UDP seja tradicionalmente usado para voz e vídeo devido à sua baixa latência. Alguns fatores podem ser citados como compatibilidade com firewalls e NAT, controle de congestionamento, confiabilidade na entrega de pacotes, buffering e adaptação em streaming, além da infraestrutura e conveniência.

Esses fatores pesam pelo fato de certas redes, especialmente corporativas, bloquearem ou limitarem o UDP por razões de segurança, enquanto permitem o TCP, que é mais amplamente aceito e tratado como tráfego "confiável".

R6 - É possível que uma aplicação tenha transferência confiável de dados mesmo quando executada sobre UDP? Se sim, como?

R: Sim, é possível que uma aplicação tenha transferência confiável de dados mesmo quando executada sobre UDP. Para isso, o desenvolvedor pode implementar mecanismos de controle de confiabilidade na camada da aplicação, simulando algumas das funcionalidades que o TCP oferece, como confirmação de recebimento (ACK), numeração de pacotes, retransmissão de pacotes perdidos, controle de fluxo e congestionamento, checksums e detecção de erros.

Estes artifícios podem transformar o UDP em uma alternativa confiável, principalmente onde se deseja controle mais granular e baixa latência.

R7 - Suponha que um processo no Host C tenha um socket UDP com o número da porta 6789. Suponha que tanto o Host A quanto o Host B enviem um segmento UDP para o Host C com o número da porta de destino 6789. Ambos os segmentos serão direcionados para o mesmo socket no Host C? Em caso afirmativo, como o processo no Host C saberá que esses dois segmentos se originaram de dois hosts diferentes?

R: Sim, ambos segmentos serão direcionados para o Host C com a porta de destino 6789, pois o UDP usa apenas o número da porta de destino para identificar o socket. Para diferenciar as origens de hosts diferentes, o processo em C necessita verificar e inspecionar as informações do cabeçalho UDP e IP sendo tais informações fornecidas pela camada de transporte. O endereço IP de origem contém o endereço de cada remetente (A e B), permitindo assim a identificação do host específico, além do número da porta de origem que ajuda a encontrar a origem do segmento caso haja múltiplas portas no mesmo host.

R8 - Suponha que um servidor Web esteja sendo executado no Host C na porta 80. Suponha que este servidor Web use conexões persistentes e esteja atualmente recebendo solicitações de dois hosts diferentes, A e B. Todas as solicitações estão sendo enviadas pelo mesmo socket no Host C? Se estiverem sendo passadas por diferentes sockets, ambos os sockets têm a porta 80?

R: Não, as solicitações de cada host (A e B) não estão sendo processadas pelo mesmo socket no Host C. No caso de conexões TCP persistentes, o servidor Web no Host C cria um socket separado para cada conexão com um cliente, mesmo que ambos usem a porta 80 como porta de destino. Isso significa que os sockets que são criados para conexões de A e B tem a porta 80 como porta de destino no servidor, mas são diferenciados pelo par IP de origem dos clientes. Assim, o servidor consegue distinguir e manter conexões usando a mesma porta 80 de destino.

Seção 3.5

R14. Verdadeiro ou falso?

a. O Host A está enviando ao Host B um arquivo grande por meio de uma conexão TCP. Suponha que o Host B não tenha dados para enviar ao Host A. O Host B não enviará confirmações ao Host A porque não pode anexar as confirmações em dados.

R: Falso, as confirmações (ACKs) são enviadas mesmo que o receptor não tenha dados para enviar de volta. O TCP usa confirmações para garantir a confiabilidade da transmissão, e isso ocorre independentemente de haver ou não dados anexados.

b. O tamanho do TCP rwnd nunca muda ao longo da duração da conexão.

R: Falso, pois o tamanho do rwnd pode mudar durante uma conexão TCP, e ele depende da quantidade de espaço livre no buffer do receptor. Quando o buffer é esvaziado (dados são processados), o valor de rwnd pode aumentar novamente.

c. Suponha que o Host A esteja enviando ao Host B um arquivo grande por meio de uma conexão TCP. O número de bytes não reconhecidos que A envia não pode exceder o tamanho do buffer de recepção.

R: Verdadeiro, pois o TCP utiliza o valor de rwnd (receiving window) para limitar a quantidade de dados não confirmados que podem ser enviados, de modo a evitar o estouro do buffer de recepção.

d. Suponha que o Host A esteja enviando um arquivo grande ao Host B por meio de uma conexão TCP. Se o número de sequência de um segmento desta conexão for m, então o número de sequência do segmento subsequente será necessariamente m + 1.

R: Falso, pois o número de sequência de um segmento TCP representa o primeiro byte de dados no segmento. Assim, o número de sequência do próximo segmento será (m + quantidade de bytes do segmento), e não necessariamente m + 1.

e. O segmento TCP possui um campo em seu cabeçalho para rwnd.

R: Verdadeiro, pois o cabeçalho do segmento TCP possui um campo chamado "Janela de Recepção" (rwnd) que indica o espaço disponível no buffer de recepção do receptor.

f. Suponha que o último SampleRTT em uma conexão TCP seja igual a 1 seg. O valor atual de TimeoutInterval para a conexão será necessariamente ≥ 1 seg.

R: Verdadeiro, pois o valor de TimeoutInterval geralmente é baseado em uma média dos valores de SampleRTT e algum desvio padrão, resultando em um valor que é pelo menos o RTT mais recente, para garantir que não ocorra timeout prematuramente.

g. Suponha que o Host A envie um segmento com número de sequência 38 e 4 bytes de dados por uma conexão TCP para o Host B. Nesse mesmo segmento, o número de reconhecimento é necessariamente 42.

R: Falso, pois os ACKs dependem do número de sequência que o Host B está esperando receber de volta, e não do número de sequência enviado. O ACK reconhece o próximo byte que espera receber, mas não depende diretamente do número de bytes enviados por A no segmento anterior.

R15. Suponha que o Host A envie dois segmentos TCP em sequência para o Host B por uma conexão TCP. O primeiro segmento tem o número de sequência 90; o segundo tem o número de sequência 110.

a. Quantos dados há no primeiro segmento?

R: O número de sequência do segundo segmento é 110, e o número de sequência do primeiro é 90. O número de sequência indica o primeiro byte de dados de cada segmento. Portanto, o primeiro segmento contém 110 - 90 = 20 bytes, pois o cálculo é baseado na diferença do primeiro segmento com o segundo.

b. Suponha que o primeiro segmento seja perdido, mas o segundo segmento chegue a B. Na confirmação que o Host B envia ao Host A, qual será o número de reconhecimento?

R: Como o primeiro segmento foi perdido, o Host B ainda espera o número de sequência 90, que corresponde ao primeiro byte do primeiro segmento que não chegou. Assim, na confirmação enviada ao Host A, o número de reconhecimento será 90, indicando que o Host B ainda está aguardando os dados a partir do byte 90.

Seção 3.7

R17. Suponha que duas conexões TCP estejam presentes em algum link de gargalo com taxa R bps. Ambas as conexões têm um arquivo enorme para enviar (na mesma direção sobre o link de gargalo). Qual taxa de transmissão o TCP gostaria de dar a cada uma das conexões?

R: Como o TCP é projetado para compartilhar a largura de banda de forma justa entre as conexões que competem por um link de rede, logo como a taxa é R, então cada conexão teria uma taxa de transmissão ideal de $\frac{R}{2}$ bps, pois o TCP tenta dividir a capacidade do link igualmente entre as duas conexões.

R18. Verdadeiro ou falso? Considere o controle de congestionamento no TCP. Quando o temporizador expira no remetente, o valor de ssthresh é definido para metade de seu valor anterior.

R: Verdadeiro, pois quando ocorre um timeout, o remetente interpreta isso como um sinal de congestionamento na rede. Assim, o protocolo TCP reduz a taxa de envio para evitar sobrecarregar a rede.