Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Redes de Computadores I
AD2 - 1° semestre de 2008.
Gabarito

Atenção: Como a avaliação a distância é individual, caso seja constatado que provas de alunos distintos são cópias uma das outras, independentemente de qualquer motivo, a todas será atribuída a nota ZERO. As soluções para as questões podem sim, ser buscadas por grupos de alunos, mas a redação final de cada prova tem que ser individual.

---

1. Suponha que a camada de rede oferece o seguinte serviço para a camada de transporte: a camada de rede no *host* de origem aceita da camada de transporte, um segmento de no máximo 1.200 bytes e o endereço do *host* de destino. A camada de rede garante a entrega do segmento na camada transporte do *host* de destino. Suponha que diversos processos de aplicações de rede podem estar em execução no *host* de destino.

a. Descreva um protocolo simples para a camada de transporte, que receberá o dado da aplicação no *host* destino. Assuma que o o sistema operacional no host destino atribui o número de porta de quatro bytes para cada processo de aplicação em execução. **(0,5 pontos)**

Resposta:

Chamemos este protocolo de Protocolo de Transporte Simples (PTS). No lado remetente, o PTS aceita do processo remetente um conjunto de dados que não excede a 1196 *bytes*, um endereço do *host* destino e um número de porta destino. O PTS acrescenta um cabeçalho de quatro *bytes* a cada conjunto de dados e põe o número da porta do processo de destino neste cabeçalho. O PTS repassa o endereço do *host* destino e o segmento resultante então à camada de rede. A camada de rede entrega o segmento ao PTS do *host* destino. O PTS examina o número da porta no segmento, extrai os dados do segmento e repassa os dados ao processo identificado pelo número da porta.

b. Modifique o protocolo de modo que ele forneça o “endereço de retorno” ao processo de destino. **(0,5 pontos)**

Resposta:

O segmento, neste caso, tem dois campos no cabeçalho: um campo de porta origem e outro para a porta destino. No lado de remetente, o PTS aceita do processo remetente um conjunto de dados que não excede a 1192 *bytes*, um endereço de *host* destino, um número da porta origem e um número da porta destino. O PTS cria um segmento que contém os dados de aplicação, número

da porta origem e número da porta destino. O segmento e o endereço do *host* destino são repassados à camada de rede. Depois de receber o segmento, o PTS do *host* destino entrega ao processo de aplicação os dados e o número porta do processo origem. A camada de rede fornece o endereço do *host* de origem.

c. Nos seus protocolos a camada de transporte "precisa executar alguma tarefa" no núcleo da rede de computadores (isto é, nos roteadores)? **(0,5 pontos)**

Resposta:

Não, a camada de transporte não tem que fazer qualquer coisa no núcleo da rede. A camada de transporte "reside" nos *hosts*, ou seja, fora do núcleo da rede.

2. Explique para que serve o processo de demultiplexação realizado na camada de transporte. **(0,5 pontos)**

Resposta:

Um processo (como parte de uma aplicação de rede) pode ter um ou mais *sockets*, portas pelas quais dados passam da rede para o processo e do processo para a rede. A tarefa de entregar dados contidos em um segmento da camada de transporte para a porta correta é denominada demultiplexação. Para tanto, na extremidade receptora, a camada de transporte examina os campos do cabeçalho do segmento para identificar a porta receptora e direciona o segmento para o *socket* correto.

3. O UDP e o TCP usam complemento de 1 para suas somas de verificação (verificação de erros). É possível que o erro de um bit não seja detectado? E um erro em dois bits? Explique. **(0,5 pontos)**

Resposta:

Considere os seguintes três bytes: 01010101, 01110000 e 01001100. Vamos então determinar o complemento de 1 para as somas desses bytes (Observe que, embora o UDP e o TCP usem *words* de 16 bits no cálculo da soma de verificação, nessa questão você deve considerar parcelas de 8 bits).

```
  0 1 0 1 0 1 0 1
  0 1 1 1 0 0 0 0
+ -----------------
  1 1 0 0 0 1 0 1

  1 1 0 0 0 1 0 1
  0 1 0 0 1 1 0 0
+ -----------------
  0 0 0 1 0 0 0 1
```

No exemplo acima o complemento a um é 1 1 1 0 1 1 1 0.
Para detectar erros, o receptor soma os quatro *bytes* (os três *bytes* originais e o *checksum*). Se a soma contiver um zero, o receptor sabe houve um erro. Todos os erros de um *bit* são assim detectados, mas erros de dois *bits* podem não ser detectados. Isto pode ser verificado, ainda no exemplo, se o último dígito do primeiro *byte* for alterado para 0 e o último dígito do segundo *byte* for alterado para 1.

4. Suponha um processo que executa no hospedeiro C tem um *socket* com número de porta 6789. Suponha que dois hospedeiros A e B, enviem segmentos UDP para a porta de destino 6789 do hospedeiro C. Responda: ambos os segmentos serão direcionados para o mesmo *socket* no hospedeiro C? Se sua resposta é sim, como o processo que executa no hospedeiro C, sabe que esses dois segmentos têm origem em dois hospedeiros diferentes? **(0,5 pontos)**

Resposta:

Sim, ambos os segmentos serão direcionados para o mesmo *socket* no hospedeiro C. Para cada segmento recebido, na interface do *socket*, o sistema operacional disponibilizará para o processo o endereço IP de forma a determinar as origens dos segmentos individuais.

5. Suponha que um servidor Web executa no hospedeiro C e escuta na porta 80. Suponha que esse servidor Web usa conexões persistentes e que está nesse momento, recebendo requisições de dois hospedeiros A e B diferentes. Explique: essas requisições são direcionadas para o mesmo *socket* no hospedeiro C? Se as requisições são direcionadas para *sockets* diferentes, podem esses diferentes *sockets* ter o mesmo número de porta 80? **(0,5 pontos)**

Resposta:

Primeira pergunta: Não. Para cada conexão persistente, o servidor de Web cria um *socket* de conexão separado. Cada *socket* de conexão é identificado pela 4-tupla <endereço IP de origem, número de porta de origem, endereço IP de destino, número de porta de destino >. Quando hospedeiro C recebe um datagrama IP, ele examina esses quatro campos nos cabeçalhos das camadas de rede e de transporte (datagrama/segmento) para determinar para qual *socket* deve passar a carga útil (*payload*) do segmento TCP.

Segunda pergunta: Como vimos na resposta à primeira pergunta, cada *socket* de conexão é identificado por quatro informações: <(IP origem, número da porta de origem, IP destino e número da porta destino>. Quando o hospedeiro C recebe um datagrama IP, este examina estes quatro campos no datagrama/segmento para determinar para qual *socket* o conteúdo de informação do segmento TCP deve ser repassado. Assim, as requisições de A e B vão para *sockets* diferentes. O identificador para ambos *sockets* tem 80 como porta de destino, porém, os identificadores para estes *sockets* têm valores diferentes para o endereço IP de origem. Diferentemente do que ocorre no UDP, quando a camada de transporte repassa o conteúdo de informações de um segmento de TCP para o processo da aplicação, não especifica o IP de origem, pois isto é implicitamente especificado pelo identificador do *socket*.

6. Descreva por que um desenvolvedor de uma aplicação distribuída escolhe executar sua aplicação sobre UDP em vez de executa-la sobre TCP. **(0,5 pontos)**

Resposta:

A escolha do UDP como o protocolo da camada de transporte é baseada em:

- **A aplicação não necessita da transferência de dados confiável:** Algumas aplicações suportam que certo percentual de dados sejam perdidos e portanto o uso do UDP é indicado.
- **Taxa de envio não regulada**: O desenvolvedor da aplicação não quer sua aplicação fique sujeita ao controle de congestionamento realizado pelo TCP que pode, em tempo de congestionamento, reduzir a taxa de transmissão da

aplicação. No TCP existem os mecanismos de controle de congestionamento e de fluxo que regulam a taxa de envio de segmentos, sem a interferência da aplicação transmissora. Este controle na taxa de envio de segmentos pode ser crítica para aplicações que requisitam uma taxa de transmissão mínima e que são tolerantes a perda de pacotes até um certo nível.

- **Não há estabelecimento de conexão:** No TCP existe o *three way handshake* antes que a transferência de dados seja iniciada.
- **Não há estado de conexão:** O TCP mantém o estado de conexão nos sistemas finais de origem e destino. Esse estado inclui *buffers* de envio e recepção, parâmetros de controle de congestionamento e parâmetros numéricos de seqüência e de reconhecimento, desta forma esta manutenção do estado tem custos de processamento e espaço alocado.
- **Pequeno *overhead* no cabeçalho do pacote:** O segmento TCP tem 20 *bytes* de cabeçalho, enquanto o do UDP tem somente 8 *bytes*.

7. Considere o protocolo Retorne a N (*Go Back N*) para transferência de dados confiável. Considere ainda que a faixa de números de seqüência é: **0, 1, 2, ..., k-1**. Qual o maior tamanho de janela possível para que o protocolo não falhe?

(1,0 ponto)

Resposta:

No protocolo Retorne a N o tamanho da janela do receptor tem por definição tamanho 1 (isto é, o receptor só recebe o segmento em seqüência). Considerando o espaço dos números de seqüência é de **k** diferentes números (isto é, os números vão de 0 até k-1), o tamanho máximo da janela do transmissor tem que ser no máximo igual a **k-1**. Suponha por exemplo, k=8 e portanto os números de seqüências possíveis são 0, 1, ..., 7. Isto nos levaria a imaginar que o tamanho da janela do transmissor pode ser 8 (isto é, o transmissor pode enviar 8 segmentos, numerados de 0 até 7, sem ter recebido qualquer confirmação). Imagine então, que o transmissor tem janela de tamanho 8 e que ele envia os 8 segmentos, e recebe a confirmação para cada um deles individualmente ou cumulativamente para todos eles. Tendo recebido essas confirmações, o transmissor avança sua janela, podendo enviar oito novos segmentos reutilizando os números de seqüência de 0 até 7. Imagine então que esses oito novos segmentos são perdidos e que o receptor envia uma duplicata de confirmação para o último segmento recebido corretamente em seqüência, isto é, o receptor confirma mais uma vez o segmento 7 da primeira leva de segmentos. O transmissor não tem como distinguir que trata-se de uma duplicata da confirmação e considera-a como sendo a confirmação que está aguardando para a nova leva de 8 segmentos enviados e não confirmados, descartando então os oito segmentos que não foram entregues ao receptor. Então o protocolo falha (o protocolo com tamanho de janela igual a 8, não faz transferência de dados confiável)! Desse modo, o tamanho da janela do transmissor pode ser no máximo 7, isto é, **k-1. Sugestão: repita o raciocínio considerando a janela do transmissor de tamanho 7, e certifique-se que assim o protocolo não falha!**

8. Considere o protocolo Repetição Seletiva (*Selective Repeat*) para transferência de dados confiável. Considere ainda que a faixa de números de seqüência é: **0, 1, 2, ..., k-1**. Qual o maior tamanho de janela possível para que o protocolo não falhe?

(1,0 ponto)

Resposta:

Tamanho máximo da janela para que o protocolo não falhe é igual a **k / 2**. Como na Questão 7, Suponha por exemplo, k=8 e portanto os números de seqüências possíveis são 0, 1, ..., 7. Admita os tamanho das janelas do transmissor e do receptor igual a 7 (podemos tentar esse tamanho de janela já, que esse valor nos garante que o protocolo Retorne a N não falha). Nesse caso, na janela do receptor são aguardados os segmentos 0, 1. 2, ...,6. Imagine então, que o remetente envia os segmentos 0, 1, 2, ..., 6 e todos eles são recebidos no destino e confirmados, passando o receptor a aguardar sete novos segmentos, os segmentos 7, 0, 1, ..., 5. Porém por uma infeliz coincidência, todas as confirmações enviadas são perdidas! Após a temporização o remetente re-envia o segmento 0, que chega ao destino e o receptor recebe essa cópia como sendo um novo dado. Então o protocolo falha (o protocolo com tamanho de janela igual a 7, não faz transferência de dados confiável)! Para o protocolo funcionar corretamente, o tamanho da janela pode ser no máximo 4, isto é, **k / 2.** Na verdade, para que o protocolo não falhe, o conjunto dos números de seqüência de uma janela e a da janela seguinte, devem ter intercessão vazia. **Sugestão: repita o raciocínio considerando a janela do transmissor de tamanho 6 e 5 e certifique -se que nos dois casos o protocolo continua falhando. Finalmente repita o raciocínio considerando a janela do transmissor de tamanho 4 e verifique que desse modo, o protocolo não falha!**

9. Considere o protocolo Retorne a N (*Go-Back-N*) com tamanho de janela do transmissor igual a 3 e uma faixa de números de seqüência de 1024. Suponha que no tempo **t** o pacote que o receptor está esperando, tenha o número de seqüência **k**. Suponha também que o meio não reordene as mensagens. Responda:

a. Quais são os possíveis de números de seqüência dentro da janela do transmissor no tempo **t** ? Justifique sua resposta. **(1,0 ponto)**

Resposta:

Temos nesse caso o tamanho da janela **N=3**. Suponha que o receptor tenha recebido o pacote **k-1** e tenha confirmado todos os outros pacotes anteriores. Se todas essas confirmações foram recebidas no transmissor, então a janela do transmissor contém pacotes com números de seqüência no intervalo **[k, k+N-1]**. Suponha agora que nenhuma dessas confirmações tenha sido recebida no transmissor. Nesse segundo, caso a janela do transmissor contém o pacote **k -1**, **k - 2** e **k - 3**. Desse modo a janela do transmissor contém pacotes com números de seqüência no intervalo **[k-N, k-1]**. Considerando essas duas situações, a janela do transmissor tem tamanho 3 e o pacote mais antigo contido na janela tem números de seqüência contidos no intervalo **[k-N, k]**.

b. Quais os possíveis valores do campo ACK na mensagem que está correntemente se propagando de volta para o transmissor no tempo t? Justifique sua resposta. **(1,0 ponto)**

Resposta:

Se o receptor está aguardando pelo pacote com o número de seqüência k, obrigatoriamente ele recebeu (e confirmou) o pacote com o número de seqüência k-1 e os N–1 pacotes recebidos antes desse. Desse modo, se nenhum desses N ACKs tiverem sido recebidos no transmissor, então as mensagens de ACK com valores no intervalo [k–N, k–1] ainda estão se propagando na rede em direção ao transmissor. Como o transmissor enviou os pacotes no intervalo [k–N, k–1], necessariamente ele já recebeu o ACK para o pacote k-N-1. Como o receptor já enviou o ACK para o pacote k-N-1, ele não

enviará confirmações que sejam inferiores a k-N-1. Assim a faixa de números das confirmações em trânsito vai de k-N-1 até k-1.

10. Imagine que o hospedeiro A envie ao hospedeiro B, por uma conexão TCP, um segmento contendo 16 bytes de dados e com número de seqüência 90. Nesse mesmo segmento, o número contido no campo de confirmação é obrigatoriamente 106? Explique sua resposta. **(0,5 pontos)**

Resposta:

A resposta é não. Os valores iniciais para os números de seqüência para cada um dos lados da conexão TCP são definidos durante o estabelecimento da conexão TCP (*three way handshake*), e cada uma das partes faz a escolha independentemente. Portanto o campo de confirmação, ao longo a transmissão na conexão TCP, reflete essa escolha inicial.

11. Os objetivos do controle de fluxo e do controle de congestionamento realizados pelo TCP, não são os mesmos. Qual o objetivo do controle de fluxo? De que forma é realizado o controle de fluxo? Qual o objetivo do controle de congestionamento? De que forma é realizado o controle de congestionamento? **(0,5 pontos)**

Resposta:

O controle de fluxo visa impedir que um receptor fique sobrecarregado por estar recebendo pacotes a uma taxa superior à que este possa consumi-los. Já controle de congestionamento visa proteger a rede de uma carga de pacotes superior a sua capacidade.

O controle de fluxo serve para evitar que o transmissor da conexão TCP envie mais dados que a capacidade de recepção do receptor dessa conexão TCP. Para tal, o transmissor precisa conhecer o espaço disponível no *buffer* de recepção do lado receptor da comunicação. Este dado é fornecido através do campo "Receive Window", que está presente no cabeçalho do TCP e que indica o espaço, em *bytes* , disponível no *buffer* de recepção. Como o TCP é *full duplex,* o campo "Receive Window" é preenchido, no lado receptor da conexão, todas as vezes que um segmento é enviado para o lado transmissor. Com o mecanismo de controle de fluxo, um problema poderia ser ocasionado quando a janela de recepção disponível chegasse a zero. Nessa situação, o transmissor não enviaria dados para o receptor e caso o receptor não tivesse dados para enviar na conexão, o transmissor não teria como saber que o espaço na janela de recepção foi liberado. O TCP resolve este problema forçando o envio periódico de pacotes, por parte do transmissor, com apenas um *byte* de dados, quando a capacidade da janela de recepção chega a zero.

Já o controle de congestionamento tem como alvo a infra-estrutura de comunicação da Internet, que interliga os dois hospedeiros, e tem como objetivo evitar um colapso de comunicação no interior da rede IP. O mecanismo de controle de congestionamento no TCP, que é baseado no "Aumento Aditivo, Diminuição Multiplicativa" (AIMD), mantém a conexão TCP em dois estados:

- **Inicio lento (*slow start* – SS):** A conexão TCP está neste estado quando a janela de congestionamento for inferior ao limiar (*threshold)*. A cada confirmação (ACK) recebida em seqüência o tamanho da janela de congestionamento é acrescido do tamanho de um MSS, o que resulta na duplicação da janela de congestionamento a cada RTT (*Round Trip Time*).
- **Prevenção de congestionamento (*congestion avoidance* - CA):** A conexão TCP está neste estado quando a janela de congestionamento (CongWin) for igual ou superior ao limiar (*threshold*). A cada confirmação (ACK) recebida em

seqüência o tamanho da janela de congestionamento é acrescido através da fórmula:

CongWin = CongWin + MSS x MSS/CongWin

A aplicação desta fórmula resulta no acréscimo do tamanho de um MSS ao tamanho da janela de congestionamento a cada RTT (*Round Trip Time*).

Obviamente, este crescimento na janela de congestionamento é efetuado de forma que não infrinja controle de fluxo.

Outros eventos, além da recepção de um ACK esperado e em seqüência, são utilizados neste mecanismo:

- Quando um ACK é recebido em duplicata é incrementado o contador de ACKs em duplicata e quando o terceiro ACK em duplicata é recebido à janela de congestionamento é reduzida à metade e o limiar (*threshold*) também recebe este mesmo valor. Portanto, a conexão entra no estado de prevenção do congestionamento.
- Quando o temporizador expira (*timeout*) o limiar recebe o valor da metade do tamanho da janela de congestionamento e a janela de congestionamento é reduzida para seu tamanho mínimo, ou seja, de um MSS. Portanto, a conexão TCP entra na fase de inicio lento (*Slow Start*).

12. Como é escolhido o valor do temporizador de uma conexão TCP? **(0,5 pontos)**

Resposta:

Denominamos “RTT amostra” (*Round Trip Time* amostra), o tempo transcorrido desde a entrega do segmento para o IP até o momento em que é recebida a sua confirmação. A maioria das implementações de TCP faz uma única medição “RTT amostra” por vez (isto é, para apenas um dos segmentos transmitidos e ainda não confirmados). As medidas obtidas para os “RTT amostra” de diferentes segmentos, variam devido a congestionamento nos roteadores e a variação de carga nos hospedeiros. Por causa dessa variação qualquer valor medido para “RTT amostra” pode ser atípico. Portanto para estimar um valor para o RTT típico, é preciso determinar algum tipo de média sobre os valores “RTT amostra” obtidos. Para tal. o TCP mantém uma média, denominada “RTT estimado” dos valores “RTT amostra”. Essa média é atualizada sempre que um novo valor de “RTT amostra” é obtido, da seguinte forma:

RTT_estimado = (1 - a) * RTT_estimado + a * RTT_amostra

Este cálculo fornece a média exponencial ponderada, na qual a influência de cada “RTT_amostra” diminui exponencialmente com o tempo. Um valor típico usado para **“a”** é 0,125.

Para determinar o valor da temporização de uma conexão TCP, ao “RTT_estimado” é adicionada uma margem de segurança. Esta margem de segurança é calculada através do desvio da amostra (denominado “desvio RTT”) em relação ao “RTT estimado”. Assim“desvio RTT” é determinado da seguinte forma:

desvio_RTT = (1 - ß) * desvio_RTT + ß * | RTT_amostra - RTT_estimado |

O valor recomendado para ß é 0,25. Com os valores de “RTT estimado” e de “desvio RTT”, o valor do temporizador é calculado da seguinte forma:

Temporizador = RTT_estimado + 4 * desvio_RTT

13. Discorra sobre o mecanismo “Aumento Aditivo, Diminuição Multiplicativa” (*Additive-Increase, Multiplicative-Decrease* - AIMD) usado no controle de congestionamento realizado pelo TCP. **(0,5 pontos)**

Resposta:

A idéia usada no controle de congestionamento do TCP é que o remetente reduza sua taxa de transmissão (isto é, reduzindo o tamanho de sua janela) quando ocorre um evento de perda. A abordagem da “diminuição multiplicativa”, reduz à metade o valor da janela do remetente após um evento de perda de segmento. Esse processo se repete sempre um evento de perda é detectado, até que a janela tenha o valor mínimo de 1 MSS (*Maximum Segment Size*). No TCP sua taxa de envio é aumentada, quando percebe que não há congestionamento, isto é, quando chegam confirmações para dados que ainda não tinham sido confirmados anteriormente. O TCP então aumenta sua taxa de envio lentamente, aumentando sua janela em 1 MSS sempre que recebe uma nova confirmação. Em resumo o TCP aumenta sua taxa de envio aditivamente quando percebe que o caminho fim-a-fim não está congestionado e a reduz multiplicativamente quando detecta (por meio de um evento de perda) que o caminho está congestionado.