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 2012.

GABARITO

---

1. **(1,0 ponto cada protocolo)** Considere os protocolos Retorne a N e Repetição Seletiva. Suponha que o número de sequência usado por ambos tenha k bits. Qual é o maior tamanho de janela para que cada um desses protocolos, de modo que transferência de dados confiável não falhe. Explique sua resposta.

   RESPOSTA:

   De modo a garantir que o protocolo Retorne a N não falhe, é necessário determinar o tamanho adequado da janela do transmissor, já que a janela do receptor nesse protocolo tem tamanho igual a 1. Considere em um primeiro momento, que o tamanho da janela do transmissor é 2^k. O transmissor então envia 2^k pacotes com números de sequência 0, 1, ... ,(2^k – 1). Considere que todos são recebidos e confirmados isoladamente (ou cumulativamente). O transmissor então, descarta as cópias dos pacotes confirmados, e envia novos pacotes com números de sequência 0, 1, ... ,(2^k – 1). Todos eles são recebidos e confirmados, porém todas as confirmações são perdidas. Nesse caso o pacote esperado na janela do receptor, tem número de sequência 0. Após o esgotamento da temporização, o transmissor, retransmite o pacote com número de sequência 0, isto é, uma duplicata, que é aceita pelo receptor como sendo um novo pacote: o protocolo falha! (note que todos os demais pacotes contidos na janela do transmissor, também são retransmitidos e recebidos pelo receptor como pacotes novos). Para garantir que o protocolo não falhe, e necessário que o tamanho da janela do transmissor, tenha no máximo, tamanho de (2^k – 1). Por exemplo, se k = 3, os 8 números de sequência usados pelo protocolo, são: 0, 1, ..., 7. Para que o protocolo não falhe, o tamanho máximo da janela do transmissor é (2^k – 1) = (2^3 – 1) = 7. Nesse caso inicialmente o transmissor envia os pacotes 0, 1, 2, 3, 4, 5 e 6. Todos são recebidos e confirmados e o receptor passa a esperar o pacote com número de sequência 7. O transmissor então envia os pacotes 7, 0, 1, 2, 3, 4, e 5, todos são recebidos e confirmados, mas todas as confirmações são perdidas. O receptor nesse momento aguarda o pacote 6. Após o esgotamento da temporização, o transmissor retransmite os pacotes 7, 0, 1, 2, 3, 4, e 5 que não são aceitos pelo receptor, pois ele aguarda o pacote com número de sequência 6.

De modo a garantir que o protocolo Repetição Seletiva não falhe, é necessário determinar a faixa de números de sequência suficiente grande, para que em qualquer instante de tempo, possa-se garantir que as janelas do transmissor e do receptor, contenham números de sequência, tais que duplicatas não sejam aceitas no receptor. Suponha que as janelas do transmissor e do receptor tenham tamanho $(2^k - 1)$. Considere então que o transmissor envia os pacotes com números de sequência de 0, 1, ... $(2^k - 2)$. A janela o receptor permite que pacotes de 0 até $(2^k - 2)$ sejam aceitos. Todos os pacotes são recebidos e confirmados pelo receptor, que passa a aguardar na sua janela os pacotes com números de sequência de $(2^k - 1)$, 0, 1, ..., $(2^k - 3)$. Porém, todas as confirmações enviadas pelo receptor são perdidas. Após o esgotamento da temporização, o transmissor retransmite o pacote com número de sequência 0, que é aceita pelo receptor: o protocolo falha!. A essência do problema é que, depois que o receptor avançou a janela, a nova faixa de números de sequência, tem intercessão com a faixa de números de sequência da janela anterior. Para garantir que o protocolo não falhe, e necessário que o tamanho da janela do receptor, tenha no máximo tamanho de $(2^k / 2)$. Por exemplo se k = 3, os 8 números de sequência usados pelo protocolo, são: 0, 1, ..., 7. Para que o protocolo não falhe, o tamanho máximo da janela do transmissor é $(2^k / 2) = (2^3 / 2) = 4$. Nesse caso inicialmente o transmissor envia os pacotes 0, 1, 2 e 3. Todos são recebidos e confirmados e o receptor passa a esperar o pacote com número de sequência 4, 5, 6 e 7. O transmissor então envia os pacotes 4, 5, 6 e 7, todos são recebidos e confirmados, mas todas as confirmações são perdidas. O receptor nesse momento aguarda os pacotes 0, 1, 2 e 3. Após o esgotamento da temporização, o transmissor retransmite os pacotes 4, 5, 6 e 7 que não são aceitos pelo receptor, pois ele aguarda pacotes com números de sequência 0, 1, 2 e 3.

2. **(1,0 ponto)** Tanto o TCP como o UDP utilizam o complementos de 1 para suas somas de verificação (*checksums*). Considere os seguintes três *words* de 16 bits: 0110 0110 0110 0000, 0101 0101 0101 0101 e 1000 1111 0000 1100. Qual o complemento de 1 para as somas desses *words*?
   RESPOSTA:
   Somando as duas primeiras *words*:

```
  0110 0110 0110 0000
+ 0101 0101 0101 0101
---------------------
  1011 1011 1011 0101
```

   Somando o resultado da soma das duas primeiras *words* com a terceira *word*:

```
  1011 1011 1011 0101
+ 1000 1111 0000 1100
---------------------
  0100 1010 1100 0001
```

(ocorre *overflow*, é preciso somar 1 ao resultado)

   Somando 1 (0000 0000 0000 0001) ao resultado da soma anterior:

```
  0100 1010 1100 0001
+ 0000 0000 0000 0001
---------------------
  0100 1010 1100 0010
```

(resultado final da soma)

   O complemento de um é o valor que somado ao resultado final da soma, produz

```
  1111 1111 1111 11111
```

. Portanto:

```
  0100 1010 1100 0010
```

(resultado da soma)

   1011 0101 0011 1101 (é o complemento de 1 procurado)

3. **(1,0 ponto)** No protocolo rdt3.0 do nosso livro texto, os pacotes de confirmação que fluem do destinatário para transmissor não têm números de sequência (embora tenham um campo de confirmação isto é para o ACK, que contém o número de sequência do pacote que estão confirmando). Por que esses pacotes de confirmação não requerem números de sequência?
   RESPOSTA:
   Números de sequência são necessários apenas para que o remetente possa ser informado pelo receptor que houve duplicidade no envio de um pacote. No caso de ACKs, o remetente não precisa desta informação, uma vez que no rdt3.0, o remetente, ao receber um ACK original, realiza sua transição para o próximo estado, ignorando assim ACKs que por ventura cheguem de forma duplicada.

4. **(1,0 ponto cada item)** No protocolo para transferência de confiável de dados Repetição Seletiva, chamamos de *MaxSeq* = $2^n - 1$ (sendo n é o número de bits do campo que transporta o número de sequência). Apesar dessa condição ser desejável para tornar a utilização dos bits do *header* mais eficiente, não provamos que ela é necessária.

4.1. Por exemplo, se n = 3, e *MaxSeq* = 4 (isto é, o protocolo utiliza os números de sequência: 0, 1, 2, 3, 4, 0, 1, 2...) o protocolo funciona corretamente? Explique sua resposta.

RESPOSTA:

Considerando os números de sequência (0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3, ...) e com MaxSeq = 4

Vamos inicialmente considerar o tamanho de janela igual a 3. Nesse caso, vamos supor que os pacotes de 0 a 2 são transmitidos, recebidos e reconhecidos pelo destinatário. A janela do receptor, passa então a aguardar os pacotes com os números de sequência (3, 0, 1). Caso, as confirmações sejam perdidas, o remetente irá temporizar e retransmitir o pacote com número de sequência 0, e este será aceito pelo receptor como um pacote contendo novos dados (e não uma duplicata). O destinatário não tem como saber se ocorreu uma retransmissão ou se trata-se da recepção de um pacote contendo novos dados. Portanto, o protocolo não funciona.

Vamos agora considerar o tamanho de janela igual a 2. Nesse caso, vamos supor que os pacotes de 0 a 1 são transmitidos, recebidos e reconhecidos pelo destinatário. A janela do receptor, passa então a aguardar os pacotes com os números de sequência (2, 3). Caso, as confirmações sejam perdidas, o remetente irá temporizar e retransmitir o pacote com número de sequência 0, e este não será aceito pelo receptor como um pacote contendo novos dados, já que na sua janela não está sendo esperado pacote com número de sequência 0. Portanto, o protocolo funciona, se as janelas do transmissor e do receptor forem dimensionadas corretamente.

4.2. Considerando os valores do item 1, qual deve ser o tamanho máximo das janelas do transmissor e do receptor? Explique sua resposta.

RESPOSTA:

É necessário garantir que números de sequência contidos em uma janela e na janela subsequente não tenham intercessão, para garantir que o problema relatado inicialmente no item 1, não ocorra. Ou seja, o tamanho máximo da janela, para os números de sequência (0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3, ...) deve ser:.

(MaxSeq + 1)/2 = $\lfloor (4+1)/2 \rfloor$ = 2 (isto é, piso de (4+1)/2)

5. **(1,0 ponto)** Explique para que serve o processo de demultiplexação realizado na camada de transporte.

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.

6. **(0,5 ponto cada item)** 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:
   6.1. Ambos os segmentos serão direcionados para o mesmo *socket* no hospedeiro C?
   RESPOSTA:
   Sim, ambos os segmentos serão direcionados para o mesmo *socket* no hospedeiro C.
   6.2.Se sua resposta é sim, como o processo que executa no hospedeiro C, sabe que esses dois segmentos têm origem em dois hospedeiros diferentes?
   RESPOSTA:
   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.

7. **(1,0 ponto)** Para que serve o campo "Janela de Recepção" (ou "RcvWindow") no cabeçalho do segmento TCP?
   RESPOSTA:
   O campo "RcvWindow" indica a quantidade de *bytes* disponível no "*buffer* de recepção" no lado receptor de uma conexão TCP. O conteúdo desse campo é usado para evitar que o transmissor envie mais dados do que o receptor é capaz de receber, evitando assim o descarte de dados, por falta de espaço de armazenamento no lado receptor de uma conexão TCP.

8. **(1,0 ponto)** Como é escolhido o valor do temporizador de uma conexão TCP?
   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 "RTTamostra" 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