UFU 45 ANOS

Lista 3 - RCI

Redes de Comunicações I

Leonardo Vecchi Meirelles

12011ECP002

Outubro 2023

R4) Muitos aplicações se adaptam melhor ao UDP pelos seguintes razões:

- Melhor controle no nível da aplicação sobre quais dados são enviados e quando;
- Não há estabelecimento de conexão;
- Não há estados de conexão;
- Pequeno excesso de cabeçalho de pacote.

R6) Sim, é possível uma aplicação ter transferência confiável de dados usando UDP. Isso pode ser feito se a confiabilidade for embutida na própria aplicação. Porém, não é trivial e exige mais do desenvolvedor. Tal processo permite que a aplicação tire proveito de ambos as alternativas, podendo se comunicar de maneira confiável sem ter de se sujeitar às limitações de taxas de transmissão impostas pelo mecanismo de controle de congestionamento do TCP.

R7) Sim, os dois segmentos serão encaminhados para o mesmo socket no host C. Para cada segmento, o sistema operacional provê os endereços IPs para que os processos determinem a origem de tal segmento

R15) O primeiro segmento possui tamanho de 20 bytes, já que possui nº de segmento 90 e o próximo segmento tem número 110, que é o nº do primeiro byte desse segmento. Ou seja, 110 - 90 = 20 bytes.

b) Como o primeiro segmento foi perdido, mesmo após a entrega do segundo, o destinatário enviará um ACK para o pacote mais recente recebido na ordem correta, nesse caso, 90.

P3) 1ª palavra - 0 1 0 1 0 0 1 1

2ª palavra - 0 1 1 0 0 1 1 0 +

1 0 1 1 1 0 0 1

3ª palavra - 0 1 1 1 0 1 0 0 +

1 0 0 1 0 1 1 0 1

1 +

soma: 0 0 1 0 1 1 1 0

Checksum: 1 1 0 1 0 0 0 1

Para detectar erros, o destinatário soma as três palavras com o Checksum. Se a soma conter um zero, o destinatário sabe que houve um erro. Todos os erros de 1 bit serão detectados, mas um erro de 2 bits pode ser não detectado.

P25) a) No TCP, um segmento pode conter mais ou menos do conteúdo de uma mensagem, ao contrário do UDP que coloca em seu segmento, qualquer coisa que a aplicação passar, independente do tamanho. Logo, com UDP, a aplicação tem maior controle de qual data é enviada ao segmento.

b) No TCP, por conta do controle de fluxo e de congestionamento, podem existir atrasos entre o momento que a aplicação envia os dados e o momento que eles são passados à camada de rede. No UDP esses controles não existem e, consequentemente, os atrasos também não.

P26) Considerando 2^{32} possíveis números de sequência;

a) O número de sequência não incrementa com cada segmento, mas com o número de bytes enviados. Logo, o tamanho de MSS é irrelevante e o máximo valor de L é simplesmente o número de bytes representáveis, nesse caso: $2^{32} \approx 4,29$ Gbytes.

b) O número de segmentos é de: $\frac{2^{32}}{536} \cong 8012999$.

66 bytes de cabeçalho são adicionados a cada segmento, logo, são $66 \cdot 8012999 = 528\,857\,934$ bytes de cabeçalho.

Assim, o total de bytes transmitidos são: $2^{32} + 528857934 = 4824 \cdot 10^6$ bytes.

Em um enlace de 155 Mbps, o tempo para transmissão é:

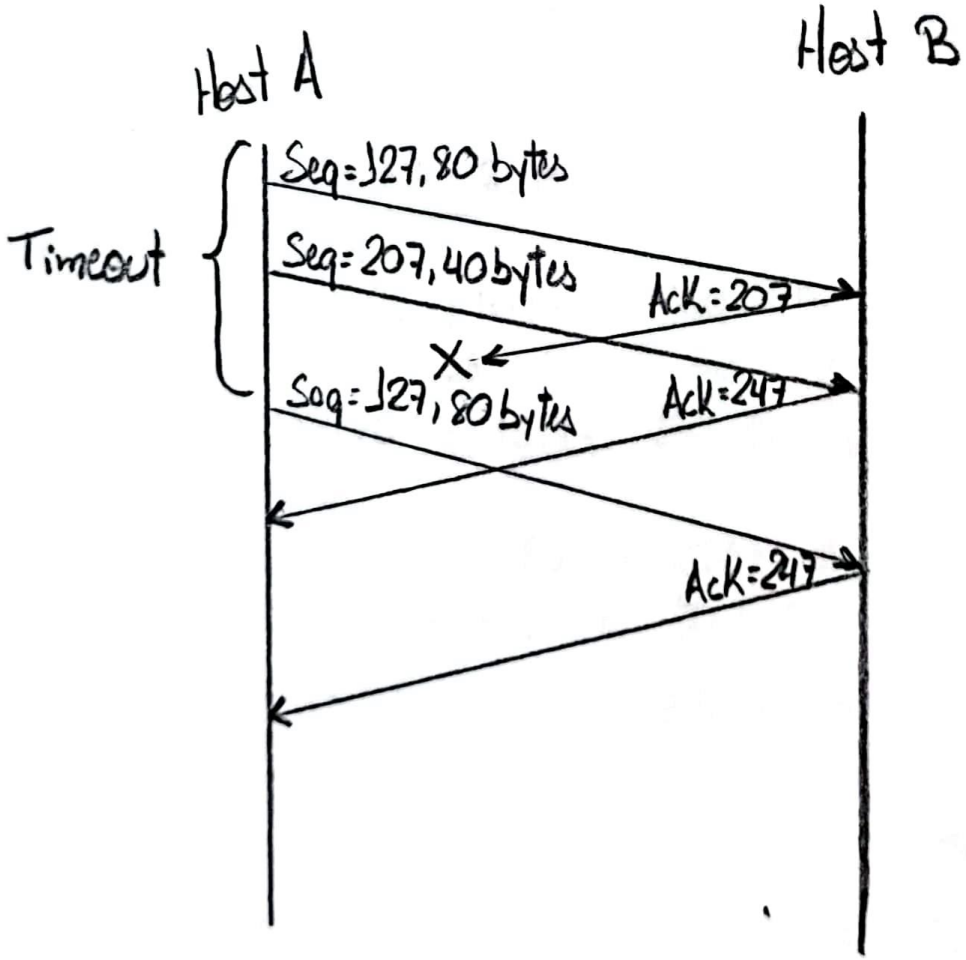
$$\frac{4824 \cdot 10^6 \cdot 8}{155 \cdot 10^6} = 248,98 \text{ segundos}$$

P27) a) O segundo segmento do host A para o B possui número de sequência 207 (127+80) e não há mudança nos portas ou seja, porta de origem 302 e de destino 80.

b) Com o primeiro segmento chegando antes, tudo está normal e o número de reconhecimento será 207. Porém, agora a porta de origem é 80 e de destino 302.

c) Caso o segundo segmento chegue antes do primeiro, o número de reconhecimento permanece em 127, indicando a espera dos bytes de 127 em diante.

d)



P31)

$EstimatedRTT = (1-\alpha)\cdot EstimatedRTT + \alpha \cdot SampleRTT$

$DevRTT = (1-\beta)\cdot DevRTT + \beta \cdot |SampleRTT - EstimatedRTT|$

$TimeoutInterval = EstimatedRTT + 4 \cdot DevRTT$

Após obter o primeiro SampleRTT = 106 ms, considerando EstimatedRTT de 100 ms e DevRTT de 5 ms, temos:

$DevRTT = 0{,}75 \cdot 5 + 0{,}25 \cdot |106 - 100| = 5{,}25 \text{ ms}$

$EstimatedRTT = 0{,}875 \cdot 100 + 0{,}125 \cdot 106 = 100{,}75 \text{ ms}$

$TimeoutInterval = 100{,}75 + 4 \cdot 5{,}25 = 121{,}75 \text{ ms}$

Com SampleRTT = 120 ms:

$DevRTT = 0{,}75 \cdot 5{,}25 + 0{,}25 \cdot |120 - 100{,}75| = 8{,}75 \text{ ms}$

$EstimatedRTT = 0{,}875 \cdot 100{,}75 + 0{,}125 \cdot 120 = 103{,}16 \text{ ms}$

$TimeoutInterval = 103{,}16 + 4 \cdot 8{,}75 = 138{,}16 \text{ ms}$

Com SampleRTT = 140 ms:

$DevRTT = 0{,}75 \cdot 8{,}75 + 0{,}25 \cdot |140 - 103{,}16| = 15{,}77 \text{ ms}$

$EstimatedRTT = 0{,}875 \cdot 103{,}16 + 0{,}125 \cdot 140 = 107{,}76 \text{ ms}$

$TimeoutInterval = 107{,}76 + 4 \cdot 15{,}77 = 170{,}84 \text{ ms}$

Com SampleRTT = 90 ms:

$DevRTT = 0{,}75 \cdot 15{,}77 + 0{,}25 \cdot |90 - 107{,}76| = 16{,}27 \text{ ms}$

$EstimatedRTT = 0{,}875 \cdot 107{,}76 + 0{,}125 \cdot 90 = 105{,}54 \text{ ms}$

$TimeoutInterval = 105{,}54 + 4 \cdot 16{,}27 = 170{,}62 \text{ ms}$

Com SampleRTT = 115 ms:

$DevRTT = 0{,}75 \cdot 16{,}27 + 0{,}25 \cdot |115 - 105{,}54| = 14{,}57 \text{ ms}$

$EstimatedRTT = 0{,}875 \cdot 105{,}54 + 0{,}125 \cdot 115 = 106{,}72 \text{ ms}$

$TimeoutInterval = 106{,}72 + 4 \cdot 14{,}57 = \underline{165 \text{ ms}}$ //

P40)

a) A partida lenta do TCP está em execução nos intervalos: [1,6] e [23,26]

b) A prevenção de congestionamento do TCP está em execução nos intervalos: [6,16] e [17,22]

c) A perda foi reconhecida por três ACKs duplicados, já que, se fosse por timeout, o tamanho da cwnd seria levado à 1.

d) Dessa vez foi reconhecida por timeout já que cwnd foi para 1.

e) O threshold é inicialmente 32, já que é o tamanho de cwnd no qual a partida lenta para.

f) Com a perda na rodada 16, quando cwnd era de 42, ssthresh é setado para metade desse valor, logo, 21.

g) Novamente, com a perda na rodada 22, quando cwnd era de 29, ssthresh foi setado para metade desse valor: 14 (arredondando para baixo).

h) Na primeira rodada, o 1° segmento é enviado. Na segunda, o 2° e o 3°. Na terceira, do 4° ao 7°. Na quarta, do 8° ao 15°. Na quinta, do 16° ao 31°; na sexta, do 32° ao 63°; na sétima, do 64° ao 96°. Logo, o 70° segmento foi enviado na setima rodada.

i) Após uma perda detectada por 3 ACKs duplicados, o tamanho da janela será reduzido ao novo threshold + 3, o qual será a metade do tamanho da janela antes da perda (8). Logo, o novo ssthresh será 4 e o tamanho da janela será 7.

j) Ssthresh ainda será 21 e, na rodada 17 o tamanho será 1; 2 na 18 e 4 na 19

K) Rodada 17: 1; 18: 2; 19: 4; 20: 8; 21: 16; 22: 21. Logo, são 52 pacotes.

P55)

a) O servidor enviará sua resposta para o endereço Y.

b) Sim, o servidor terá certeza de que o cliente estará no endereço Y. Caso existisse outro endereço falsificando Y, o SYSNACK seria enviado ao endereço Y e o TCP no host não enviaria o TCP ACK de volta. Mesmo que o invasor mandasse um TCP ACK no momento correto, ele ainda não saberia o número de sequência do servidor certo, já que o servidor usa um número de sequência aleatório.