Resolução Lista 3

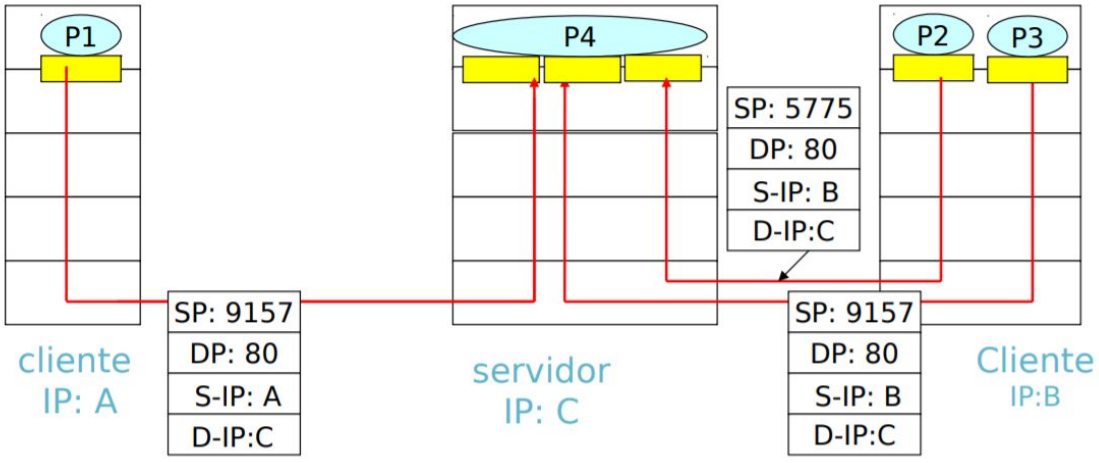
Mateus Carvalho Gonçalves - 201810245
Otávio Augusto de Sousa Resende - 201810543

1) O TCP possui um campo “window” para controle de fluxo. Através deste campo o software TCP indica quantos dados ele tem capacidade de receber em seu buffer.

2) Os protocolos TCP e UDP não dispõem de serviços para garantia de banda e de atrasos.

3) O TCP utiliza de serviços de rdt (reliable data transfer) para cobrir a transmissão não confiável do IP. Mais especificamente, o TCP utiliza o envio paralelo de segmentos com reconhecimento por ACKs cumulativos, e dois casos disparam a necessidade de retransmissão dos segmentos: quando recebe ACKs duplicados e quando há timeout de um evento de tempo de confirmação.

4) A web implementa o protocolo TCP na camada de transporte e esse cria sockets diferentes para cada conexão de clientes diferentes e de processos distintos de um mesmo cliente. Os sockets são identificados por 4 campos: porta de destino, porta de origem, IP de destino e IP de origem. Exemplo com valores nos campos (slide 12 do material cedido pelo professor)



5) Na transmissão de pacotes podem haver perdas, pois com frequência o tempo de expiração é relativamente longo, gerando longos atrasos antes de reenviar o pacote perdido. Essas perdas podem ser percebidas pelo recebimento de ACKs duplicados. Com isso o protocolo TCP utiliza-se da retransmissão rápida que consiste no reenvio do segmento antes de o temporizador expirar.

6) a. Falso
b. Falso

c. Verdadeiro. Obs: a dupla entendeu “tamanho do buffer de recepção” como o tamanho disponível do buffer, ou seja, (tamanho total - tamanho em uso).

d. Falso

e. Verdadeiro

f. Falso

g. Falso

7) 1° segmento, de A -> B, Seq = 43, ACK = 80; Dados = "R";
2° segmento, de B -> A Seq = 80, ACK = 44, Dados = “R”;
3° segmento, de A -> B, Seq = 44, ACK = 81, Dados = nada.

8) No primeiro caso o envio de NAK é adequado. Apesar de a detecção de perda ser mais lenta, o envio de dados é esporádico e isso provê mais tempo para recuperação. Já no segundo caso, a perda só seria notificada após o envio de vários pacotes (com reconhecimento NAK), então o ACK se torna mais viável, principalmente quando há pipelining.

9) A resposta dos itens *a* e *b* são semelhantes, por isso foram aglutinadas em apenas 1. O serviço UDP apenas recebe os dados de um processo, os encapsula e envia para a camada de rede. Essa característica implica que ele não implementa controle de congestionamento. Dessa forma, a aplicação obtém mais controle de quais e quando os segmentos são enviados.

10) Fórmulas:
$EstimatedRTT = \alpha * SampleRTT + (1-\alpha) * EstimatedRTT$
$DevRTT = \beta * |SampleRTT - EstimatedRTT| + (1 - \beta) * DevRTT$
$TimeoutInterval = EstimatedRTT + 4 * DevRTT$

Primeiro:
$EstimatedRTT = 0,125 * 106 + 0,875 * 100 = 100,75$ ms
$DevRTT = 0,25 * |106 - 100,75| + 0,75 * 5 = 5,06$ ms
$TimeoutInterval = 100,75 + 4 * 5,06 = 120,99$ ms

Segundo:
$EstimatedRTT = 0,125 * 120 + 0,875 * 100,75 = 103,15$ ms
$DevRTT = 0,25 * |120 - 103,71| + 0,75 * 5,06 = 8$ ms
$TimeoutInterval = 103,15 + 4 * 8 = 135,15$ ms

Terceiro:
$EstimatedRTT = 0,125 * 140 + 0,875 * 103,15 = 107,76$ ms
$DevRTT = 0,25 * |140 - 107,76| + 0,75 * 8 = 14,06$ ms
$TimeoutInterval = 107,76 + 4 * 14,06 = 164$ ms

Quarto:

EstimatedRTT = 0,125 * 90 + 0,875 * 107,76 = 105,54 ms
DevRTT = 0,25 * |90 - 105,54| + 0,75 * 14,06 = 14,42 ms
TimeoutInterval = 105,54 + 4 * 14,06 = 163,22 ms
Quinto:
EstimatedRTT = 0,125 * 115 + 0,875 * 105,54 = 106,71 ms
DevRTT = 0,25 * |115 - 106,71| + 0,75 * 14.42 = 12,88 ms
TimeoutInterval = 106,71 + 4 * 12,88 = 158,23 ms

11) a. O número de sequência no TCP incrementa pelo número de bytes transmitidos, então o tamanho do MSS é irrelevante para o limite máximo do tamanho do arquivo. Esse limite é dado pelo número de bytes representável por 2^{32} (tamanho do campo de número de sequência).

b. 2^{32} / 536 = 8.012.999 segmentos
8.012.999 * 66 = 528.857.934 bytes
2^{32} + 528.857.934 ≈ 4.8GB
4.8GB / (155/8)MB ≈ 250s