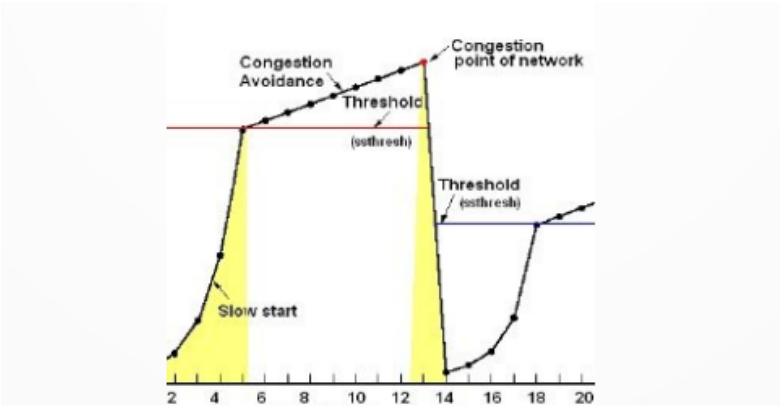
Redes 2

O Protocolo TCP - Continuação

- Controle de congestionamento
  - Slow-Start TCP
    - Um dos algoritmos de controle de congestionamento do TCP
    - Inicialmente JCong = 1 MSS (Maximum Segment Size)
    - MSS = 1480 bytes de TCP (mais 20 bytes de header IP)
    - Transmite JCong = 1 ……. ACK
      Dobra JCong = 2 ……ACK
      Dobra JCong = 4 ……ACK
      Dobra JCong = 8 ……TIMEOUT
      Neste momento, considera que a rede está congestionada
      Volta a JCong = 1
      Continua …
    - Marca um limite (threshold) de metade da JCong que levou ao congestionamento da rede
    - A partir deste ponto limite, o Slow-Start incrementa a JCong de 1 em 1



    -
  - Timeout
    - A base é o RTT (Round Trip Time)
    - Tempo entre uma mensagem ir da origem ao destino e a resposta chegar do destino à origem
    - O TCP começa a medir no estabelecimento da conexão (SYN - 3HS)
    - O TCP, na verdade, faz uma média ponderada do RTT, pois existe (normalmente) uma pequena variação
    - $RTT_{médio} = \alpha * RTT_{medio} + (1-\alpha) * RTT_{novo}$
    - No linux, $\alpha = 0,9$ (em geral, 0,1)
    - Se faz a média ponderada para se “desconsiderar” variações extremas
    - O Timeout [Original] = $\beta * RTT_{medio}$ onde originalmente $\beta = 2$
    - Porém, a média não reflete a dispersão
    - O Desvio Médio é uma maneira melhor de levar em conta a dispersão (não se usa desvio padrão clássico, pois é caro de calcular)
    - $Desvio = |RTT_{medio} - RTT_{novo}|$
    - $Desvio_{medio} = \alpha * Desvio_{medio} + (1-\alpha) * Desvio$
    - Portanto, o timeout = $RTT_{medio} + \beta * Desvio_{medio}$ onde $\beta = 4$

    - O Slow Start “puro” é implementado pelo TCP Tahoe
    - O TCP Reno adotou a chamada “retransmissão rápida”
        - Considere o seguinte caso, quando 1 pacote e perde, no Tahoe o efeito é terrível na taxa, reduzindo drasticamente a janela
        - Portanto, quando recebe 3 ACKs para um mesmo segmento antigo, existe uma alta probabilidade de que o seguinte se perdeu
        - Então, retransmite o que foi depois, sem esperar o timeout, evitando a redução da JCong
- Algoritmo de Nagle
    - Evita que o TCP mande muitos pacotes pequenos
    - Usado para melhorar o uso da rede
    - Aumenta a taxa dados_payload/dados_controle
    - Ideia: vai juntando pequenas quantidades para fazer uma transmissão maior
    - Algoritmo
        - O primeiro byte manda direto
        - Depois, só manda se
            - Tiver recebido ACK para todas as transmissões
            - Ou tem ½ da janela
            - Ou tem MSS bytes para transmitir
- Solução de Clark
    - Mesmo raciocínio, do lado do receptor
    - Espera receber vários dados, para não ficar mando 1 ACK por byte
    - Deste modo, um ACK só é transmitido
        - Para ½ do buffer a confirmar
        - Ou MSS bytes a confirmar
        - Ou um Timer de limite de espera