P1 - 19 de Setembro de 2011

ACH2026 - Redes de Computadores (Valdinei Freire da Silva)

Nome: _____ NUSP: _____

1. Indique se cada uma das afirmações a seguir é verdadeira ou falsa e **dê uma justificativa** curta (entre uma e duas linhas) para sua resposta. Respostas com a justificativa incorreta não serão consideradas.
   a) Enlaces por rádio, além de serem mais flexíveis por não exigirem a instalação de fios, sempre são tão confiáveis quanto enlaces de par trançado, cabo coaxial ou fibra ótica. [0.5]
   b) É possível que um protocolo X, da camada A da pilha de protocolos da Internet, realize transferência confiável de dados usando um protocolo não-confiável Y da camada B imediatamente inferior. [0.5]
   c) HTTP é um protocolo que não armazena estado e, por isso, é impossível para um servidor HTTP modificar o conteúdo que envia para o cliente com base no estado de sua navegação. [0.5]
   d) O protocolo UDP fornece garantia que as mensagens entregues não tem erro. [0.5]
2. Considere um hospedeiro A enviando um único pacote de tamanho L bytes para o hospedeiro B. A e B encontram-se a uma distância de 25Km e estão conectados por meio de 5 enlaces de 5Km cada, nos quais a velocidade de propagação do sinal é de 2.5×10^8m/s, totalizando 4 roteadores no caminho. Todos os enlaces tem capacidade de transmissão de 10^6 bytes/s (∼8Mbps). Todos os roteadores, bem como hospedeiros, demoram 10μs para processar o pacote. Considere que A é o único hospedeiro enviando pacotes pela rede no momento e que a mesma não está congestionada, não havendo atraso de enfileiramento.
   a) Calcule o atraso, em função de L, para enviar o pacote de A a B usando **Comutação de Pacotes**, com todos os roteadores utilizando transmissão store-and-forward (ou seja, o pacote deve ser recebido por inteiro antes de começar a ser enviado). [0.75]
   b) Calcule o atraso, em função de L, para enviar o pacote de A a B usando **Comutação de circuitos**, sem atraso devido ao store-and-forward ou processamento, mas com necessidade de 4.25ms para estabelecer a conexão e imaginando que todos os recursos são dedicados para essa conexão. [0.75]
   c) Analisando os resultados obtidos nos itens anteriores, é possível concluir que, nesse cenário, para pacotes com até um certo tamanho L1, o uso da comutação de pacotes é vantajosa, apesar do atraso de store-and-forward, devido ao tempo necessário para estabelecer a conexão, mas para pacotes maiores a comutação de circuitos acaba fornecendo um atraso menor. Calcule o valor de L1. [0.5]
   d) Imagine agora que, no caso da comutação de circuitos, os recursos não são mais dedicados à conexão entre A e B, mas estão igualmente divididos, usando TDM, por um total de N ($1<N\leq4$) possíveis conexões, mesmo que a conexão entre A e B continue sendo a única ativa no momento. Recalcule L1 em função de N. [0.5]
3. DNS
   a) Servidores de DNS possuem ao menos 4 tipos de registro: CNAME, A, NS e MX. Os registros funcionam como mapeamento entre dois tipos de informação, dentre as quais: nome de servidores, domínios e endereços IP. Defina o mapeamento realizado por cada tipo de registro. [0.5]
   b) Servidores de DNS são organizados hierarquicamente em servidores: raiz, TLD, autoridade e locais. Considere uma empresa que registrou o domínio redes.com.br. Esta empresa possui um servidor web, um servidor de e-mail e um servidor de nomes (autoridade) conforme as informações abaixo:

Servidor	Endereço IP	Nome
E-mail	143.137.2.22	mail.redes.com.br
Web	143.137.2.23	web.redes.com.br
DNS	143.137.2.24	dns.redes.com.br

   Considere ainda que tal empresa deseje que usuários acessem seu servidor web utilizando o nome www.redes.com.br. Considere os servidores TLD e de autoridade envolvidos e indique os registros necessários em cada um deles para

que clientes possam acessar os servidores dessa empresa. [1.0]

c) Uma mensagem DNS possui 1 seção para enviar perguntas e possui 3 seções para transportar registros DNS: *answer*, *authority* e *additional*. Quando um usuário envia uma pergunta a respeito do servidor web da empresa anterior para seu servidor de nomes local, quais são as trocas de mensagens entre o servidor local do usuário e os outros servidores envolvidos. Considere que o servidor de nomes local utilize mensagens no modo iterativo. Note que não é obrigatório utilizar a seção *additional*, mas se a mesma for utilizada, a quantidade de mensagens trocadas pode ser reduzida. [1.0]

4. Considere uma implementação do TCP com os seguintes eventos:

   - **TimeOutX:** indica que temporizador X esgotou.
   - **RecSeg:** indica que um segmento foi recebido.
   - **ReadData(Length):** indica que a camada de aplicação requisitou a leitura de *Length* bytes do buffer.
   - **SendData(Data):** indica que a camada de aplicação requisitou o envio dos dados *Data* para o destinatário.

   Para simplificação, considere que um segmento TCP possui apenas os seguintes campos autoexplicativos: *Sequence*, *Ack*, *Window*, *Length* e *Data*. Considere o seguinte pseudo código utilizado para implementar o TCP quando a conexão já foi estabelecida, isto é, o handshake de três vias já ocorreu. Note que existem vários pontos não tratados explicitamente no código, ele apenas serve como regra para em que momentos enviar os segmentos. Por exemplo, a quantidade de números de sequência disponíveis é infinita.

```
%Constantes
MAXBUFFER = 20 %Tamanho máximo do Buffer Entrada
MSS = 5 %Quantidade máxima de dados em um segmento

%Variáveis para controle do buffer de Saída
NxtSeq = 1 %Número de sequência do próximo segmento
SndBase = 1 %Número de sequência ainda não reconhecido
NxtData = 1 %Próximo byte a ser recebido da aplicação
SndWnd = MAXBUFFER %Janela de envio

%Variáveis para controle do buffer de Entrada
RcvWnd = MAXBUFFER %Janela de recebimento
NxtRead = 1 %Próximo byte a ser lido pela aplicação
LastAck = 1 %Próximo byte a ser recebido
NewAck = false %Existe segmento não reconhecido?

inicializa Temporizador 1

loop(forever) switch(event)

  event: SendData(Data)
    aloca os dados no buffer de saída
    NxtData = NxtData + Length(Data)

  event: ReadData(Len)
    retorna Len bytes em NxtRead do buffer de Entrada
    NxtRead = NxtRead + Len
    RcvWnd = MAXBUFFER - (LastAck - NxtRead)

  event: RecSeg(Seq,Ack,Win,Len,Data)
    aloca os dados no buffer de entrada
    if(Ack>SndBase)
      SndBase = Ack
      if(SndBase < NxtSeq)
        reinicializa o temporizador 2
      else
        para o temporizador 2
    if(Len > 0) then NewAck=true
    if(LastAck==Seq) then LastAck=Seq+Len
    RcvWnd = MAXBUFFER - (LastAck - NxtRead)
    SndWnd = Win

  event: TimeOut2
    %Este temporizador controla o reenvio de pacotes
    Len = min(MSS,SndWnd,NxtData-SndBase)
    pct = (SndBase, LastAck, RcvWnd, Len, Data)
    envia pct
    reinicializa Temporizador 1
    reinicializa Temporizador 2

  event: TimeOut1
    %Este temporizador controla a taxa de envio de pacotes
    if (NxtData>NxtSeq) e (SndWnd > NxtSeq - SndBase)
      Len = min(MSS,SndWnd-(NxtSeq-SndBase),NxtData-
NxtSeq)
      pct = (NxtSeq, LastAck, RcvWnd, Len, Data)
      envia pct
      if Temporizador 2 não está rodando
        inicializa Temporizador 2
      NxtSeq = NxtSeq + Len
    elseif(NewAck)
      pct = (NxtSeq, LastAck, RcvWnd, 0)
      envia pct
    NewAck = false
    inicializa Temporizador 1
```

a) Considere um cenário no qual o Hospedeiro A requisita à camada de transporte que envie 30 bytes, isto é, no tempo 0 ocorre o evento **send(data)** onde **data** contém 30 bytes, e o Hospedeiro B apenas recebe tais bytes. Considere ainda que os tempos utilizados para o TimeOut1 e TimeOut2 são respectivamente 10ms e 50ms e que os tempos de ida ou de volta são 20ms, isto é, RTT=40ms. Faça a simulação considerando que a aplicação no Hospedeiro B lê 3 bytes a cada 10ms. Indique claramente os segmentos trocados entre os Hospedeiros com os valores de: *Sequence*, *Ack*, *Window* e *Length*. Se houver conflito na ocorrência de eventos, trate-os na ordem que aparecem no pseudo-código. [1.5]

b) Considere o mesmo cenário acima, mas agora o TimeOut2=30ms e a aplicação no Hospedeiro B lê todos bytes disponíveis a cada 100ms. [1.5]