Fundação CECIERJ - **Vice Presidência de Educação Superior a Distância**

Curso de Tecnologia em Sistemas de Computação
Disciplina: Redes de Computadores I
Gabarito da AP2 - 1° semestre de 2010.

1. **(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.

2. **(2,5 pontos)** Suponha que um servidor *Web* é executado no **Computador C** na porta 80. Esse servidor utiliza conexões persistentes e no momento, está recebendo solicitações de dois computadores diferentes: o **Computador A** e o **Computador B**. Todas as solicitações estão sendo enviadas através do mesmo *socket* no **Computador C**? Se as solicitações estão passando por diferentes *sockets*, podem ambos os *sockets* ter o número de porta 80? Discuta e explique**.**
   Resposta:
   Para a primeira pergunta a resposta é não. Para cada conexão persistente, o servidor de Web cria um *socket* de conexão separado. Cada *socket* de conexão é identificado pela 4-tupla <endereço IP de origem, número de porta de origem, endereço IP de destino, número de porta de destino >. Quando hospedeiro C recebe um datagrama IP, ele examina esses quatro campos nos cabeçalhos das camadas de rede e de transporte (datagrama/segmento) para determinar para qual *socket* deve passar a carga útil (*payload*) do segmento TCP.
   Respondendo a segunda pergunta, como vimos na anterior, cada *socket* de conexão é identificado por quatro informações: <endereço IP origem, número da porta de origem, endereço IP destino e número da porta destino>. Quando o hospedeiro C recebe um datagrama IP, este examina estes quatro campos no datagrama/segmento para determinar para qual *socket* o conteúdo de informação do segmento TCP deve ser repassado. Assim, as requisições de A e B vão para *sockets* diferentes. O identificador para ambos *sockets* tem 80 como porta de destino, porém, os identificadores para estes *sockets* têm valores diferentes para o endereço IP de origem. Diferentemente do que

ocorre no UDP, quando a camada de transporte repassa o conteúdo de informações de um segmento de TCP para o processo da aplicação, não especifica o IP de origem, pois isto é implicitamente especificado pelo identificador do *socket*.

3. **(2,5 pontos)** O **hospedeiro A** está enviando um arquivo enorme para o **hospedeiro B** através de uma conexão TCP. Nessa conexão não há perda de pacotes e os temporizadores nunca se esgotam. Seja **R** bps a taxa de transmissão do enlace que liga o **hospedeiro A** à Internet. Suponha que o processo que executa no **hospedeiro A** é capaz de enviar dados para o seu *socket* TCP à uma taxa de **S** bps, onde **S** = 10 x **R**. Suponha ainda que o *buffer* de recepção do TCP (no **hospedeiro B**) seja grande o suficiente para conter o arquivo inteiro, e que o *buffer* de envio TCP (no **hospedeiro A**) pode conter apenas um por cento do arquivo. O que impediria o processo que executa no **hospedeiro A** de passar dados continuamente para o seu *socket* TCP a taxa de **S** bps: o controle de fluxo TCP; o controle de congestionamento TCP; ou alguma outra coisa? Explique sua resposta.

   Resposta:

   Neste problema não existe o perigo do transmissor sobrecarregar (inundar) o receptor, pois o *buffer* do receptor é capaz de armazenar o arquivo inteiro, logo o controle de fluxo não impede o **hospedeiro A** de passar dados continuamente para o seu *socket* TCP a taxa de **S** bps. Considerando que não existem perdas e que os ACKs retornam antes dos temporizadores expirarem, o mecanismo de controle do congestionamento não influi na taxa do transmissor. Contudo, o processo do **hospedeiro A** não poderá passar dados continuamente para o *socket,* ipois o *buffer* de transmissão é pequeno e logo fica cheio. Portanto, tão logo que o *buffer* de transmissão está cheio, o processo repassa os dados à uma taxa média R que é muito menor que S.

4. **(2,0 pontos)** Considere o protocolo Retorne a N para transferência confiável de dados. Suponha que o espaço de números de sequência usado é de tamanho **k** (isto é, vai de **0** até **k-1**). Responda: qual o maior tamanho de janela (do transmissor) que pode ser usado, para garantir que o protocolo não falhe? Explique sua resposta.

   Resposta:

   No protocolo Retorne a N o tamanho da janela do receptor tem por definição tamanho **1** (isto é, o receptor só recebe o segmento em sequência). Considerando o espaço dos números de sequência é de **k** diferentes números (isto é, os números vão de **0** até **k-1**), o tamanho máximo da janela do transmissor tem que ser no máximo igual a **k-1**. Suponha por exemplo, **k=8** e portanto os números de sequências possíveis são **0, 1, ..., 7**. Isto nos levaria a imaginar que o tamanho da janela do transmissor pode ser **8** (isto é, o transmissor pode enviar **8** segmentos, numerados de **0** até **7**, sem ter recebido qualquer confirmação). Imagine então, que o transmissor tem janela de tamanho **8** e que ele envia os **8** segmentos, e recebe a confirmação para

cada um deles individualmente ou cumulativamente para todos eles. Tendo recebido essas confirmações, o transmissor avança sua janela, podendo enviar oito novos segmentos reutilizando os números de sequência de **0** até **7**. Imagine então que esses oito novos segmentos são perdidos e que o receptor envia uma duplicata de confirmação para o último segmento recebido corretamente em sequência, isto é, o receptor confirma mais uma vez o segmento **7** da primeira leva de segmentos. O transmissor não tem como distinguir que trata-se de uma duplicata da confirmação e considera-a como sendo a confirmação que está aguardando para a nova leva de **8** segmentos enviados e não confirmados, descartando então os oito segmentos que não foram entregues ao receptor. Então o protocolo falha (o protocolo com tamanho de janela igual a **8**, não faz transferência de dados confiável)! Desse modo, o tamanho da janela do transmissor pode ser no máximo 7, isto é, **k-1.**

5. **(2,0 pontos)** Uma confirmação TCP perdida não necessariamente força uma retransmissão. Por que?

   Resposta:

   Por causa do uso da confirmação cumulativa. Ou seja, o recebimento da confirmação ***n*** pelo transmissor TCP, indica que todos ***n - 1*** *bytes* anteriores do fluxo de *bytes*, foram recebidos no receptor TCP. Essa confirmação ***n*** pode estar confirmando *bytes* contidos em um ou mais segmentos, cujas confirmações foram perdidas (ou não enviadas).