UNIVERSIDADE DE ÉVORA
ESCOLA DE CIÊNCIAS E TECNOLOGIA
Departamento de Informática

Redes de Computadores

Primeira Frequência - 10 de março de 2022

Parte I

1. Para cada uma das seguintes afirmações, indique se é verdadeira ou falsa.

   Em qualquer dos casos, justifique a sua resposta.

   (a) Num protocolo *Stop&Wait*, o tempo de propagação do canal influencia a escolha do tamanho da janela.

   > **Solução:** Falsa.
   > No protocolo *Stop&Wait* não existe janela, que é o mesmo que dizer que a janela tem sempre tamanho 1.

   (b) A taxa de utilização de um canal usando *Go-Back-N* com uma janela de tamanho 2 é sempre maior que a taxa de utilização do mesmo canal usando *Stop&Wait*.

   > **Solução:** Verdadeira.
   > Mantendo as mesmas condições, haverá maior taxa de utilização com uma janela de tamanho 2, comparativamente a uma janela de tamanho 1.

   (c) Supondo que estamos a usar pacotes de 200 bytes, podemos dizer que o protocolo *Selective Repeat* com janela de tamanho 5 é exactamente igual ao *Stop&Wait* com pacotes de 1000 bytes.

   > **Solução:** Falsa.
   > Apesar de o número de bytes enviados de cada vez ser o mesmo, se usarmos *Stop&Wait*, em caso de erro teremos de reenviar sempre os 1000 bytes, enquanto que com *Selective Repeat* só teremos de enviar os pacotes (de 200 bytes) que não chegaram correctamente.

   (d) Um protocolo do tipo *Stop&Wait* é útil numa rede com perdas.

   > **Solução:** Verdadeira.
   > Se há perdas, o protocolo *Stop&Wait* vai ajudar a que sejam reenviados os pacotes perdidos ou que chegam com erros.

   (e) Há algumas vantagens em usar um protocolo *Go-Back-1*[1] em vez de usar *Stop&Wait*.

[1] Janela de tamanho 1

Solução: Falsa.
Um protocolo de janela deslizante com janela de tamanho 1 é exactamente igual ao protocolo *Stop&Wait*.

(f) A taxa de utilização tende a crescer quando aumentamos o RTT.

Solução: Falsa.
O RTT está no divisor da fórmula, logo o aumento do RTT faz com que a taxa de utilização diminua.

(g) Num protocolo *Stop&Wait*, o tamanho da janela é definido consoante o tempo de propagação do canal.

Solução: Falsa.
Por lapso, esta pergunta saiu igual à (a), portanto a resposta é a mesma.

(h) A taxa de utilização de um canal usando *Go-Back-N* com uma janela de tamanho 1 é sempre maior que a taxa de utilização do mesmo canal usando *Stop&Wait*.

(i) Um bom *timeout* deve ser directamente proporcional ao tempo de transmissão.

Parte II

2. Considere um sistema de *framing* em que se usa a flag `01010` para marcar o início de cada *frame*.

(a) Proponha um sistema de *bit stuffing* e aplique-o à seguinte mensagem:

`1 1 0 0 1 0 0 1 1 0 1 0 1 1 0 1 0 1 0 1 0 0 0 1 0 1 0 1`

Solução: Uma boa ideia será evitar que o último zero da flag apareça na mensagem. Assim, a proposta será colocar um `1` de cada vez que a última sequência enviada tenha sido `0101`, evitando o aparecimento da flag, caso o próximo bit seja `0`.

Aplicando à mensagem:

`1 1 0 0 1 0 0 1 1 0 1 0 1 1 1 0 1 0 1 1 0 1 0 1 1 0 0 0 1 0 1 1 0 1`.

(b) Suponha agora que queremos marcar não só o início da frame mas também o seu fim. Para isso, usamos a *flag* `01011`, mantendo a outra no início.

Será necessário alterar o esquema de *bit stuffing* neste caso? Justifique.

Solução: Seria necessário procurar outro sistema de bit stuffing, pois ao mantermos o proposto acima, vamos transformar a flag de início na flag de fim, criando um novo problema na comunicação.

Parte III

3. Considere dois *hosts* de rede `A` e `B`, ligados por um canal de 10Mbps. `A` está a enviar pacotes de 1500 bytes para `B`. Foi executado um comando `'ping'` de `A` para `B`, devolvendo o seguinte output:

```
$ ping www.google.com
PING B (w.x.y.z) 56(84) bytes of data.
64 bytes from B (w.x.y.z): icmp_seq=1 ttl=118 time=34.7 ms
64 bytes from B (w.x.y.z): icmp_seq=2 ttl=118 time=34.2 ms
64 bytes from B (w.x.y.z): icmp_seq=3 ttl=118 time=34.0 ms
^C
--- B ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 33.977/34.282/34.685/0.297 ms
```

Desprezando o tempo de transmissão de um `ACK`, responda às seguintes questões:

(a) Qual é o número máximo de pacotes por segundo que `A` consegue transmitir para `B`, se não for usado qualquer protocolo de transporte?

Solução: Assumindo que não existe protocolo de transporte, o limite é apenas o débito do canal.

Assim, temos:

`Tamanho do pacote:` $1500bytes \times\ 8 = 12000bits$
`Débito do canal:` $10Mbps = 10 \times 10^6 bps$

$$T_T = \frac{12 \times 10^3}{10 \times 10^6} = 0.0012s$$

Por segundo conseguimos enviar:

$$\frac{1}{0.0012} \simeq\ 833pkts/s$$

(b) Se for usado o protocolo *Stop&Wait*, qual será o número **máximo** de pacotes por segundo que `A` consegue enviar para `B`?

Solução: Se usarmos o protocolo *Stop&Wait*, para cada pacote temos de esperar pelo seu `ACK` e só depois podemos enviar o próximo. Sabendo que o `RTT` é de 33.977, temos:

$$T_T + RTT = 0.035177s \simeq\ 35ms$$

Ou seja, por cada período de 35ms (+/-) só podemos enviar um pacote.

$$\frac{1}{0.035177} \simeq\ 28.42 \approx\ 28pkts/s$$

i. E o mínimo?

Solução: No resultado do `ping` apresentado só conseguimos ver uma ínfima parte dos tempos de `RTT` obtidos. Se prolongássemos o tempo, provavelmente iríamos obter tempos muito mais altos (potencialmente ∞). Apesar de não termos mais informação, podemos dizer que, no pior caso, seriam enviados `0` (zero) pacotes por segundo.

(c) Qual é a taxa de utilização do canal, nas condições da alínea anterior?

Solução:

$$T_U = \frac{0.0012}{0.035177} \simeq\ 0.0341131 \approx\ 3.4\%$$

(d) Seria útil, nestas condições, usar um protocolo *Go-Back-5*? Justifique a sua resposta.

Solução: Sim, pois aumentando a janela vamos aumentar a taxa de utilização (sem passar dos 100%, claro).

Exercício: provar que isto é verdade, calculando a taxa de utilização nesta situação.

(e) Proponha um *timeout* adequado para este canal.

Solução: Não existe apenas **uma** solução para esta questão, mas tentaremos obter um timeout que não seja muito desfasado dos tempos de $T_T + RTT$. Uma opção válida é usar o `RTT` médio apresentado no enunciado (mais T_T), outra seria usar um Δ que fosse adequado, por exemplo `2ms` (ficando o timeout em `37ms`. Interessa que não seja muito longo – por exemplo `10ms` já seria quase um terço do tempo total de trânsito do pacote.