Teoría de las comunicaciones

Manuel Mena

10 de abril de 2019

Índice

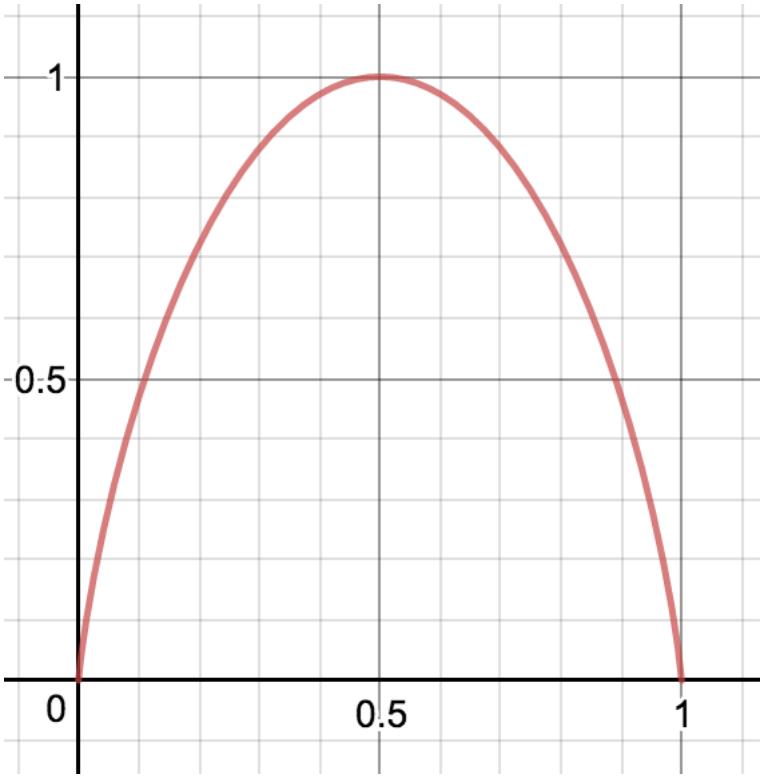
1. Práctica 1

1.1.

1.1.a.

$$
\begin{aligned}
I(s) &= -log_2(P(s)) \\
H(S) &= \sum_{s \in S}(P(s)I(s)) \\
&= (P(s_0)I(s_0)) + (P(s_1)I(s_1)) \\
&= (p_0(-log_2(p_0))) + (1-p_0)(-log_2(1-p_0))) \\
&= -p_0 log_2(p_0) - log_2(1-p_0) + p_0 log_2(1-p_0) \\
&= p_0(log_2(1-p_0) - log_2(p_0)) - log_2(1-p_0) \\
&= p_0 * log_2(\tfrac{1-p_0}{p_0}) - log_2(1-p_0)
\end{aligned}
$$

1.1.b.



1.2.

1.2.a.

El tercero

1.2.b.

El segundo y el tercero

1.2.c.

$$
\begin{aligned}
H(S) &= \sum_{s \in S}(P(s)I(s)) \\
&= 0{,}4(-log_2(0{,}4)) + 0{,}3(-log_2(0{,}3)) + 0{,}2(-log_2(0{,}2)) + 0{,}1(-log_2(0{,}1)) \\
&= 1{,}84643934467 \\
L(segundo) &= \sum_{s \in S}(P(s)|C(s)|) \\
&= 0{,}4 * 1 + 0{,}3 * 2 + 0{,}2 * 3 + 0{,}1 * 3 \\
&= 1{,}9
\end{aligned}
$$

$$\begin{aligned} L(tercero) &= \sum_{s\in S}(P(s)|C(s)|) \\ &= 0{,}4*1+0{,}3*2+0{,}2*3+0{,}1*4 \\ &= 2 \end{aligned}$$

Eficiencia segundo: 0,9718101814

Eficiencia tercero: 0,9232196723

El segundo es más eficiente.

1.2.d.

No, puesto que $H(S) \leq L(C)$ en ambos casos.

1.3.

1.3.a.

$$\begin{aligned} H(S) &= \sum_{s\in S}(P(s)I(s)) \\ &= 0{,}5(-log_2(0{,}5)) + 0{,}5(-log_2(0{,}5)) \\ &= 1 \\ L(C) &= \sum_{s\in S}(P(s)|C(s)|) \\ &= 0{,}5*1+0{,}5*1 \\ &= 1 \end{aligned}$$

1.3.b.

$$\begin{aligned} H(S) &= \sum_{s\in S}(P(s)I(s)) \\ &= 0{,}25(-log_2(0{,}25)) + 0{,}25(-log_2(0{,}25)) + 0{,}25(-log_2(0{,}25)) + 0{,}25(-log_2(0{,}25)) \\ &= 2 \\ L(C) &= \sum_{s\in S}(P(s)|C(s)|) \\ &= 0{,}25*2+0{,}25*2+0{,}25*2+0{,}25*2 \\ &= 2 \end{aligned}$$

1.3.c.

$$\begin{aligned} H(S) &= \sum_{s\in S}(P(s)I(s)) \\ &= \tfrac{1}{6}(-log_2(\tfrac{1}{6})) + \tfrac{1}{6}(-log_2(\tfrac{1}{6})) + \tfrac{1}{6}(-log_2(\tfrac{1}{6})) + \tfrac{1}{6}(-log_2(\tfrac{1}{6})) + \tfrac{1}{6}(-log_2(\tfrac{1}{6})) + \tfrac{1}{6}(-log_2(\tfrac{1}{6})) \\ &= 2{,}58496250072 \\ L(C) &= \sum_{s\in S}(P(s)|C(s)|) \\ &= \tfrac{1}{6}(-log_2(\tfrac{1}{6})) + \tfrac{1}{6}(-log_2(\tfrac{1}{6})) + \tfrac{1}{6}(-log_2(\tfrac{1}{6})) + \tfrac{1}{6}(-log_2(\tfrac{1}{6})) + \tfrac{1}{6}(-log_2(\tfrac{1}{6})) + \tfrac{1}{6}(-log_2(\tfrac{1}{6})) \\ &= 2{,}58496250072 \end{aligned}$$

1.3.d.

$$\begin{aligned} H(S) &= \sum_{s\in S}(P(s)I(s)) \\ &= 0{,}125(-log_2(0{,}125))*8 \\ &= 3 \end{aligned}$$

1.3.e.

$$\begin{aligned} H(S) &= \sum_{s\in S}(P(s)I(s)) \\ &= 0{,}1(-log_2(0{,}1))*10 \\ &= -log_2(0{,}1) \\ &= log_2(10) \\ &= 3{,}32192809489 \end{aligned}$$

1.3.f.

$$\begin{aligned} H(S) &= \sum_{s\in S}(P(s)I(s)) \\ &= \tfrac{1}{n}(-log_2(\tfrac{1}{n}))*n \\ &= -log_2(\tfrac{1}{n}) \\ &= log_2(n) \end{aligned}$$

1.4.

Acá lo mas importante es notar que cada vez que la fuente D toma un símbolo es independiente de la anterior, entonces se puede definir $P_D(s) = (P_D(A) * P_A(s) sis \in A sino P_D(B) * P_B(s))$, con $P_D(A)$ y $P_D(B)$ la probabilidad de que D saque un símbolo de A y de B, respectivamente. Entonces la solución sería:

1. $P_D(s) = 1/16$ si $s \in A$ sino $1/32$, no es equiprobable
2. $P_D(s) = 1/64$ si $s \in A$ sino $3/64$, no es equiprobable
3. $P_D(s) = 1/32$ si $s \in A$ sino $3/64$, no es equiprobable
4. $P_D(s) = 1/12$ si $s \in A$ sino $1/48$, no es equiprobable

1.5.

1.5.a.

Como la fuente es equiprobable:

$$\begin{aligned} H(S) &= log_2|S| \\ &= log_2(10^{640*480}) \\ &= 640 * 480 * log_2(10) \end{aligned}$$

1.5.b.

En esta fuente equiprobable, tenemos que $H(S) = log_2(n)$, no siendo n una potencia de 2. Esto implica que no podemos pensar en recurrir a un código (óptimo) que asigne $log_2(n)$ bits por imagen, pues por supuesto estos valores deben ser números enteros. No obstante, la aproximación más cercana a este valor es tomar exactamente $\lceil log_2(n) \rceil$ bits por imagen. De esta forma podremos armar un código que sea unívocamente decodificable (cada imagen recibe una tira de bits distinta) y localmente óptimo, entendiendo por esto que todo otro código que mapee cada imagen a una tira de bits de igual largo debe necesariamente poseer una longitud media igual o mayor. Como nota adicional, extrapolando los conceptos plasmados en la resolución de este ejercicio, es interesante pensar en si, efectivamente, dada una fuente equiprobable S' de m símbolos, y dado un código C que actúe sobre S' definiendo para cada símbolo una codificación de largo $\lceil log_2(m) \rceil$ bits, se tiene que C es óptimo. Cuando m es potencia de 2, la respuesta es afirmativa (¿por qué?). En cualquier otro caso, y quizás yendo a contramano de la intuición, la respuesta es que no. Como ejemplo sencillo, considerar el caso en el que $m = 3$. Como $\lceil log_2(m) \rceil = 2$, tenemos que $L(C) = 2$. Sin embargo, podemos proponer un código alternativo C' que asigne un 0 a un símbolo, un 10 a otro y un 11 al restante. Es claro que C' es instantáneo, y además se ve claramente que $L(C') < 2 = L(C)$

1.5.c.

Para responder este ítem, hay que usar el teorema de Shannon: $C = B * log_2(1 + SNR)$. Primero, la relación señal-ruido hay que pasarla de decibeles a veces: $SNR = 10^{30/10} = 1000$. Luego, como se necesitan 30 imágenes por segundo, la capacidad de canal debe ser mayor o igual que $\lceil 640 * 480 * log_2(10) * 30 \rceil$ bps. Entonces, $B = (\lceil 640 * 480 * log_2(10) \rceil * 30)/log_2(1001)$ Hz.

1.6.

Para calcular la capacidad de volumen, necesitamos $Delay$ y V_{tx}. Primero calculamos la Capacidad de Transmisión de Shannon para obtener la V_{tx} (dejamos la resolución para el inciso b):

$$\begin{aligned} C &= B * log_2(1 + SNR[dB]) \\ &= 400kHz * log_2(1 + 10[dB]) \\ &= 400kHz * log_2(1 + 10[veces]) \implies V_{tx} \leq 1383,7Kbps \end{aligned}$$

Ahora Delay:

$$\begin{aligned} Delay &= T_{prop} + T_{tx} \\ &= \frac{D}{V_{prop}} + \frac{|UnidaddeDatos|}{V_{tx}} \\ &= \frac{100km}{200000km/s} + \frac{1bit}{1383Kbps} = 0,0005seg = 0,5ms \end{aligned}$$

Finalmente:

$C_{vol} = Delay * V_{tx} = 691bits$

1.7.

$C_{max} = 100\text{Mbps}$

$V_{prop} = 300000$ km/s

$$\begin{aligned} Delay &= T_{tx} + T_{prop} \\ Delay &= \frac{|datos|}{V_{tx}} + \frac{D}{V_{prop}} \\ Delay &= \frac{30Mb}{100Mbps} + \frac{D}{300000km/s} \\ 0,04s &\geq \frac{30Mb}{100Mbps} + \frac{D}{300000km/s} \\ 0,04s - \frac{30Mb}{100Mbps} &\geq \frac{D}{300000km/s} \\ 0,04s - 0,3\text{s} &\geq \frac{D}{300000km/s} \\ (-0,26)s * 300000km/s &\geq D \\ -78000km/s &\geq D \end{aligned}$$

No pueden existir distancias negativas. Siendo la velocidad de transmicion de 100 Mpbs, no pueden enviarse 30 Mb en menos de 0.3 s.

1.8.

1.8.a.

$$\begin{aligned} Delay &= T_{prop} + T_{tx} \\ &= \frac{D}{V_{prop}} + \frac{|UnidaddeDatos|}{V_{tx}} \\ &= \frac{385000km}{300000km/s} + \frac{1bit}{100Mbps} \\ &= 1,29s \end{aligned}$$

$RTT = Delay * 2 = 2,58s$

1.8.b.

$$\begin{aligned} C_{vol} &= Delay * V_{tx} \\ &= 2,58s * 100Mbps \\ &= 258Mb \end{aligned}$$

1.8.c.

$$\begin{aligned} T_{total} &= Delay_{ida} + Delay_{vuelta} \\ &= T_{prop} + T_{tx_{ida}} + T_{prop} + T_{tx_{vuelta}} \\ &= \frac{D}{V_{prop}} + \frac{|UnidaddeDatos|_{i}da}{V_{tx}} + \frac{D}{V_{prop}} + \frac{|UnidaddeDatos|_{v}uelta}{V_{tx}} \\ &= \frac{2D}{V_{prop}} + \frac{|UnidaddeDatos|_{i}da + |UnidaddeDatos|_{v}uelta}{V_{tx}} \\ &= \frac{2*385000km}{300000km/s} + \frac{2Kb+25Mb}{100Mbps} \\ &= 2,57s + \frac{2Kb+25000Kb}{100000Kbps} \\ &= 2,57s + \frac{25002Kb}{100000Kbps} \\ &= 2,57s + 0,25s \\ &= 2,82s \end{aligned}$$

2. Práctica 2

2.1.

2.1.a.

$$\begin{aligned} C_{vl} &= Delay * V_{tx} \\ &= 1{,}25s * 1Mbps \\ &= 1{,}25Mb \end{aligned}$$

2.1.b.

Entran $\frac{1{,}25Mb}{1Kb} = \frac{1250Kb}{1Kb} = 1250$ Frames

2.2.

2.2.a.

$Eframe = \frac{largodelosdatos}{largodelosdatos} = 1$

2.2.b.

$Eframe = \frac{largodelosdatos}{largodelosdatos+16}$

2.2.c.

$Eframe = \frac{largodelosdatos}{largodelosdatos+8+bitsincorporadosdebidoalstuffing}$

2.3.

2.3.a.

Para el primer caso podríamos definir un frame para el emisor y otro para el receptor de la siguiente forma:

Emisor: #SEQ(1bit); Datos; Checksum

Receptor: #ACK(1bit); Checksum

2.3.b.

Para el segundo caso, al haber Piggybacking todos los frames pueden llegar a tener que cumplir simultáneamente los roles de emisión y recepción. Entonces proponemos un único frame como el siguiente:

#SEQ; #ACK; Datos; Checksum

2.3.c.

Para el último caso, a diferencia del punto anterior tenemos reconocimento selectivo. Para estos casos se recomienda, por un tema de eficiencia, que el frame receptor reconozca los frames tanto acumulativa como individualmente. Entonces nos quedarían dos frames como los siguientes:

Emisor: #SEQ; Datos; Checksum

Receptor: #ACK; #SACK; Checksum

2.5.

2.5.a.

$$\begin{aligned} SWS &= \frac{V_{tx} * RTT}{|Frame|} \\ &= \frac{1Mbps * 2 * 0{,}25s}{2Kb} \\ &= \frac{500Kb}{2Kb} \\ &= 250 \end{aligned}$$

$RWS = 1$

2.5.b.

$$\begin{aligned} \#frames \geq SWS + RWS &= \frac{V_{tx} * RTT}{|Frame|} \\ &= \frac{1Mbps * 2 * 0{,}25s}{2Kb} + 1 \\ &= \frac{500Kb}{2Kb} + 1 \\ &= 251 \end{aligned}$$

$$\begin{aligned} \#frames &\geq 251 \end{aligned}$$

Por lo tanto se necesitan $\lceil log_2(251) \rceil = 8$ bits.

2.5.c.

Frame: #SEQ (8 bits); Datos (1976 bits) CRC (16 bits) (2Kb total)

$20Mb$ de datos $= \lceil \frac{20000000bits}{1976bits} \rceil$ frames = 10,122 frames

$$\begin{aligned} Delay(10122Frames) &= Delay(10122 * 2Kb) \\ &= Delay(20122Kb) \\ &= T_{tx}(20122Kb) + T_{prop} \\ &= \frac{20122Kb}{V_{tx}} + 0{,}25s \\ &= \frac{20122Kb}{1Mbps} + 0{,}25s \\ &= 20{,}122s + 0{,}25s \\ &= 20{,}372s \end{aligned}$$

A esto se le agrega el envío de el último frame

2.6.

2.6.a.

$E_{frame} = \frac{largodelosdatos}{largototaldelframe}$

2.7.

$E_{proto} = \frac{T_{tx}(V)}{RTT(F)}$

2.8.

2.8.a.

$$\begin{aligned} Delay(F) &= T_{tx}(F) + T_{prop} \\ Delay(1Kb) &= T_{tx}(1Kb) + T_{prop} \\ 270\text{ms} &= \frac{1Kb}{1Mbps} + T_{prop} \\ 270\text{ms} &= \frac{1}{1000}\text{s} + T_{prop} \\ 270\text{ms} &= 1\text{ ms} + T_{prop} \\ 269\text{ms} &= T_{prop} \end{aligned}$$

$RTT(F) = 2 * Delay(F) = 2 * 269\text{ ms} = 538\text{ ms}$

$$\begin{aligned} E_{proto} &= \frac{T_{tx}(V)}{RTT(F)} \\ &= \frac{T_{tx}(7 * 1Kb)}{RTT(F)} \\ &= \frac{\frac{7Kb}{1Mbps}}{538ms} \\ &= \frac{\frac{7}{1000}s}{538ms} \\ &= \frac{7ms}{538ms} \\ &= 0,013 \end{aligned}$$

2.8.b.

$$\begin{aligned} E_{proto} &= \frac{T_{tx}(V)}{RTT(F)} \\ &= \frac{T_{tx}(127*1Kb)}{RTT(F)} \\ &= \frac{\frac{1277Kb}{1Mbps}}{538ms} \\ &= \frac{\frac{1277}{1000}s}{538ms} \\ &= \frac{127ms}{538ms} \\ &= 0,236 \end{aligned}$$

2.8.c.

$$\begin{aligned} E_{proto} &= \frac{T_{tx}(V)}{RTT(F)} \\ &= \frac{T_{tx}(255*1Kb)}{RTT(F)} \\ &= \frac{\frac{2557Kb}{1Mbps}}{538ms} \\ &= \frac{\frac{2557}{1000}s}{538ms} \\ &= \frac{255ms}{538ms} \\ &= 0,474 \end{aligned}$$

2.9.

2.9.a.

$$RTT(F) = 2 \text{ s}$$

$$\begin{aligned} T_{tx}(V) &= T_{tx}(8 * 2Kb) \\ &\frac{8*2Kb}{V_{tx}} \\ &\frac{8*2Kb}{10Kbps} \\ &1{,}6 \text{ s} \end{aligned}$$

$$\begin{aligned} E_{proto} &= \frac{T_{tx}(V)}{RTT(F)} \\ &= \frac{1{,}6s}{2s} \\ &= 0{,}8 \end{aligned}$$

2.9.b.

Recordar que se tiene un esquema de reconocimiento de acknowledge selectivo

1. Emisor envía el 4
2. Emisor envía el 5
3. Emisor envía el 6
4. Emisor envía el 7
5. Receptor recibe el 4 mal
6. Receptor recibe el 5, envía $ACK = 3$ $SACK = 5$
7. Receptor recibe el 6 mal
8. Receptor recibe el 7, envía el $ACK = 3$ $SACK = 7$
9. Emisor recibe ACK del 5, reenvía el 4
10. Emisor recibe ACK del 7, reenvía el 6
11. Receptor recibe el 4, envía el $ACK = 5$ $SACK = 4$
12. Receptor recibe el 6, envía el $ACK = 7$ $SACK = 6$
13. Emisor recibe ACK del 4
14. Emisor recibe ACK del 6

3. Práctica 3

3.1.

3.1.a.

$t_{min} = 2 * Delay_{max} = 2 * 25{,}6\mu s = 51{,}2\mu s$

3.1.b.

$V_{tx} = 10Mbps$
Regla de 3 simple. En un segundo transmito 10 Mb, en $51{,}2\mu s$ transmito
$51{,}2\mu s * 10Mbps = 51{,}2\mu s * 10bp\mu s = 512bits = 64B$

3.1.c.

Se rellena con padding. Hay dos opciones para luego descartar el padding:

- En el header: en lugar de usar el campo type para multiplexar se usa como length tamaño y se usa LLC como multiplexador para la capa de red.
- Se encarga la capa superior (por ejemplo IP length).

3.1.d.

Ambos sensan el medio en los respectivos momentos temporales e identifican que el medio está siendo utilizado. Ergo, esperan (1-persistente) hasta que el medio esté libre y luego transmiten. Ambos encontrarán el medio libre en momentos muy cercanos (la diferencia está dada por la distancia entre H2 y H3 con H1) y sus tramas colisionarán.

3.1.e.

H4 sensa el medio. Probablemente lo encuentre libre (por ejemplo si todavía no llega a sensar la información de la trama de H1 por estar a una distancia considerable), transmita y su trama colisione con la de H1.

3.4.

3.4.a.

El switch root es S1. Quedan bloqueados los puertos pertenecientes a L3.

3.4.b.

Se desbloquean los puertos pertenecientes a L3 y se bloquean los que conectan L1.

3.6.

3.6.a.

Recordar que el puerto del switch que queda disponible es el del que menor distancia al root tenga.

Si empatan, queda disponible el de menor id

	H5
1001	2
1002	2
1003	3
1004	3
1005	2
1006	WIFI

3.6.b.

	H5
1001	WIFI
1002	1
1003	1
1004	1
1005	1
1006	1

3.8.

3.8.b.

2

4.

4.2.

4.2.a.

Dependiendo de la version el formato del header es distinto, por lo que no saber que version es puede provocar no poder leer correctamente los campos.

4.2.c.

El tamaño máximo de un paquete IP es 65535 bytes. El campo lenght define este tamaño.

4.3.

4.3.a.

Tiene 3 interfaces: FastEthernet, Wireless0 Connection y Wirless1 Connection

4.3.b.

- Physical Address es la dirección MAC
- IP Address es la direccion IP
- Subnet Mask es la máscara de subred que se aplica
- Default Gateway es el router por defecto

4.4.

4.4.a.

Es solo pasar los /N a XXX.XXX.XXX.XXX
/22 = 255.255.252.0
/23 = 255.255.254.0
/24 = 255.255.255.0
/25 = 255.255.255.127

4.4.b.

135.46.56.0/22 = 135.46. 0011 1000 .0 / 255.255.252.1111 1100 = 135.46.56.0
Capacidad máxima de hosts esta dada por /22, son todos las direcciones que podemos meter sobre los 10 digitos que la máscara anula. Osea $2^{10} = 1024$ hosts.

4.4.c.

	A	B
135.46.57.14	Interface1	descarta
135.46.63.10	Interface0	descarta
135.46.52.2	135.46.62.100	descarta
208.70.188.15	135.46.62.100	descarta
135.46.62.62	Interface0	descarta
192.53.40.7	135.46.60.100	Interface1
192.53.56.7	135.46.62.100	descarta

4.5.

PC1: 172.16.5.

4.6.

4.6.a.

Pueden direccionarse 256 - 2 hosts.

4.6.b.

2 subredes: 128 - 2 hosts.
4 subredes: 64 - 2 hosts.
8 subredes: 32 - 2 hosts.

4.7.

4.7.a.

Red A: 172.16.5.0/27 (32)
Red B: 172.16.5.32/29 (8) Dejo 8 porque se necesitan 1 host + 2 routers + 2 de broadcast y localhost
Red C: No puedo usar 15.16.5.40/27 (128) porque se solaparía con 17.16.5.64/22 que ya existe

6. Práctica 6

6.1.

Un puerto es una interfaz a través de la cual se pueden enviar y recibir los diferentes tipos de datos.

Si no se implementaran los puertos, entonces las aplicaciones de dos hosts distintos se comunicarían de manera broadcast, ya que al enviar un dato se estaría enviando a todo quien escuche en ese host. Teniendo puertos es posible definir de donde recibir y hacia donde enviar.

UDP es mas rápido ya que hace menos chequeos pero es menos seguro y no es confiable. Es útil en caso de una aplicación de llamadas de voz o streaming.

TCP al contrario, tiene más delay, pero asegura que los datos enviados lleguen y de manera segura, y en orden. Por lo que sirve para chats, o cualquier aplicación donde es absolutamente necesario que los datos sean recibidos.

6.5.

6.5.a.

Hay 2 conexiones TCP: (20.232.1.1:80,1.2.3.4:5678) y (1.1.1.1:2222,3.3.3.3:4444)

6.5.b.

El criterio es fijarse en la dirección IP y puerto del destino y el orígen.
Para la conexión (20.232.1.1:80,1.2.3.4:5678) los segmentos: 1, 3, 4, 6, 8, 9, 11, 12 y 14
Para la conexión (1.1.1.1:2222,3.3.3.3:4444) los segmentos: 2, 5, 7, 10 y 13

6.5.c.

Puede verse que éste ocurre en el paquete 13, cuando el proceso en 3.3.3.3:4444 envía un RST. Esto se debe al paquete con flags SYN/ACK/URG recibido en la línea 10: recibir un SYN en una conexión sincronizada tiene como efecto el envío de un RST, conforme a la especificación de TCP (para más detalles, referirse a la página 71 del RFC 793). La consecuencia de este envío es que el receptor procederá a liberar los recursos de la conexión de inmediato, sin ejercer el algoritmo de cierre tradicional. A su vez, el host emisor del RST hace esto mismo en reacción al paquete mal formado que recibió.

6.5.d.

- Del paquete 1 se desprende que 1.2.3.4:5678 está haciendo una apertura activa. Por ende, este paquete hace que su TCP se mueva de CLOSED a SYN-SENT.
- El paquete 3 es una respuesta positiva del paquete 1 por parte del proceso 20.232.1.1:80. Para que esto ocurra, su TCP debe haber estado en LISTEN antes de poder contestar. Al recibir el SYN, la reacción consiste en trasladarse a SYN-RCVD y emitir un paquete SYN+ACK.
- El primer host recibe luego este segmento y esto hace que se mueva a ESTABLISHED, enviando como respuesta el ACK final del three-way handshake (paquete 4).
- Una vez procesado esto, el TCP de 20.232.1.1:80 se moverá también a ESTABLISHED. Luego tenemos que los paquetes 6, 8 y 9 corresponden a envío y reconocimiento de datos, por lo que no motivan transiciones entre estados.
- En la porción final de la captura, vemos que se inicia el proceso de cierre de conexión. El segmento 11 nos muestra que 20.232.1.1:80 está haciendo un cierre activo, revirtiendo así su rol “pasivo” al momento de establecer la conexión. El efecto de este envío es transicionar al estado FIN-WAIT-1.
- En la siguiente línea vemos que su interlocutor recibe este mensaje y reacciona respondien- do un paquete FIN+ACK que no sólo reconoce el primer FIN (puesto que su #ACK es 10202) sino que además inicia el cierre de su stream de escritura. En consecuencia, su TCP colapsa dos estados, pasando de ESTABLISHED a LAST-ACK.

- El segmento 14 reconoce correctamente el FIN de 1.2.3.4:5678. Al recibirlo, el TCP de dicho proceso transicionará a CLOSED, mientras que el de 20.232.1.1:80 se moverá a TIME-WAIT como respuesta al segmento 12. Asumiendo que el último ACK no se extravía en la red, allí permanecerá por 120 segundos (i.e., 2 MSLs) antes de pasar también a CLOSED.

6.7.

6.7.a.

- El host 2.71.82.81:823 pasa al estado FIN_WAIT_1
- El host 3.14.15.92:654 recibe el FIN y envía su ACK, pasando al estado CLOSE_WAIT. El host 2.71.82.81:823 recibe el ACK por lo que pasa al estado FIN_WAIT_2
- 3.14.15.92:654 envía FIN pasando al estado LAST_ACK
- 2.71.82.81:823 envía el último ACK, pasando al estado TIME_WAIT y provocando que 3.14.15.92:654 pase al estado CLOSED. Luego de un tiempo de espera, 2.71.82.81:823 también pasará a CLOSED.

6.10.

6.10.b.

Es luego del paquete 4, una vez recibido en SYN/ACK cuando el receptor puede comenzar a enviar datos junto con un ACK que finaliza el three-way handshake

6.11.

6.11.a.

1. 192.168.0.100 pasa de CLOSED a SYN_SENT
2. 157.92.27.21 pasa de LISTEN a SYN_RCVD
3. 192.168.0.100 pasa a ESTABLISHED al enviar el ACK y 157.92.27.21 también lo hace al recibirlo
4. Para el resto la conexión se mantiene establecida

6.12.

6.12.a.

1. 157.92.27.21:80 - 10.80.1.10:35336 SYN,ACK
2. 10.80.1.10:35336 - 157.92.27.21:80 ACK
3. 157.92.27.21:80 - 10.80.1.10:35336 FIN
4. 10.80.1.10:8080 - 200.11.54.101:32154 SYN
5. 200.11.54.101:32154 - 10.80.1.10:8080 SYN,ACK
6. 10.80.1.10:8080 - 200.11.54.101:32154 ACK

6.12.b.

$t_1 \geq t_0 + 2$ segment lifetimes

7. Práctica 7

7.1.

7.1.a.

Uno solo

7.1.b.

Una sola

7.1.c.

Uno solo

7.1.d.

También uno. El servidor tiene un puerto abierto para atender a cada cliente que haga un pedido. Una vez atendido abre otro puerto y le comunica al cliente que a partir de ahora lo va a se comunican por ese otro puerto.

7.2.

7.2.a.

GET / HTTP 1.1 Host: dc.uba.ar

7.2.b.

HEAD /tdc HTTP 1.1 Host: dc.uba.ar

7.2.c.

GET IF-MODIFIED-SINCE *unafecha* /logo.jpg HTTP 1.1 Host: www.dc.uba.ar

7.3.

7.3.a.

HTTP 1.0 -¿[syn] ¡- [syn+ack] -¿[ack] + GET index HTTP 1.0 ¡- [fin] 200 OK *index* -¿[fin+ack] ¡- [ack]

En total lo de arriba son 3 RTT. Se repite lo de arriba para:

- style.css
- searchline.png
- home.png
- search_icon.gif

Luego, en total son 5*3 = 15 RTT

7.3.b.

$-> [syn] < -[syn+ack] -> [ack] + GET index HTTP 1,1 < -200 OK * index * -> GET style.css HTTP 1,1 < -200 OK * style.css * -> GET searchline.png HTTP 1,1 < -200 OK * searchline.png * -> GET home.png HTTP < -200 OK * home.png * -> GET search_icon.gif HTTP 1,1 < -200 OK * search_icon.gif * -> [fin] < -[fin + ack] -> [ack]$

En total son 7,5 RTT.

8. Práctica 8

8.3.

Generalmente $CWND = IW = 2 * SMSS$ con $SMSS = 2KB$ y $SSTHRESH = 64KB$

8.4.

Ante un timeout se modifican $CWND = LW = SMSS = 2KB$ y $SSTHRESH = max(FLIGHT\_SIZE/2, 2 * SMSS)$

8.5.

Porque de esa forma sería casi seguro que se generara congestión.

8.6.

Debe incrementarse un $SMSS$ por RTT durante Congestion Avoidance.