

# CC51C Comunicación de Datos

## Control 1

(con apuntes)

Profesor: Jens Hardings Perl  
Prof. Auxiliar: Sebastián Castro  
4 de Mayo de 2001

De las preguntas 1 a la 3, elija solamente 2.

### P1

i) Un alumno opina que es poco práctico estar calculando el checksum en cada hop, y que se debiera calcular un checksum ignorando ese campo. Con esto, se cumpliría mejor el principio de "la inteligencia en las puntas" de IP.

(a) ¿Cómo implementaría este esquema?

*suponiendo que ese campo tiene algún valor fijo (ej: 0) para efectos de cálculo del checksum, igual como se ignora el checksum al calcular el checksum. También se podría usar un pseudo-header que simplemente no incluya ese campo, o bien mover ese campo fuera del header (por ejemplo al final del paquete).*

(b) ¿Cuáles serían los efectos positivos?

*Los routers no se preocuparían de volver a calcular el checksum, aumentando así teóricamente el ancho de banda que pueden manejar, o bien serían más simples al no necesitar la implementación en hardware del algoritmo.*

*Originalmente, el checksum se calcula dos veces: primero para revisar el checksum del paquete que recién llega y luego se recalcula el checksum del paquete saliente, que tiene distinto TTL. Sería correcto suponer además que no se calcula el primer checksum tampoco, con lo cual la eficiencia sería mayor (pero esto tiene otros efectos negativos que se deben considerar).*

(c) ¿Cuáles serían los efectos negativos?

*Para que los efectos positivos sean reales, no se revisa el checksum en ningún router intermedio. Por ello, es posible que se alteren algunos bits algún campo sin que se detecte hasta el final, donde sí se revisa el checksum. Esto básicamente significa trabajo extra de ruteo que pudo haberse evitado.*

*Si se revisa el checksum original en los routers intermedios, lo único que puede cambiar sin ser detectado es el TTL mismo, lo cual en el peor de los casos lleva a descartar erróneamente paquetes, o bien demorarse más en descartar paquetes en loop.*

- ii) Se propone que el algoritmo de ruteo en IP instale en la tabla de rutas la dirección física del gateway en vez de la dirección IP. ¿Opiniones?

*Es una mala idea. Aunque me ahorro el ARP para traducir dirección IP a física la primera vez, pierdo la independencia de la red física en el nivel ruteo. Por otro lado, no me permite cambios de tarjetas físicas ni reemplazar un router por otro sin cambiar todas las tablas en la red. Pueden existir otros problemas, como direcciones físicas duplicadas en dos redes conectadas a mi router (no en ethernet, pero sí en otras tecnologías), por lo que requiero además que diga la interfaz por la que debe llegarse a esa dirección.*

- iii) Dé un ejemplo en el cual un esquema de tipo *Go-Back-N* se comporte mejor que uno como *Selective Repeat*. (Mejor comportamiento en el sentido de enviar menos bytes totales para una misma cantidad de información útil enviada).

*Se envían  $n$  paquetes seguidos ( $n > 1$ ), y el receptor envía los  $n$  acks. Los primeros  $n-1$  acks se pierden, pero el  $n$ -ésimo llega. El algoritmo *Go-Back-N* capta que llegaron todos los paquetes, mientras que el *Selective Repeat* debe retransmitir los  $n-1$  paquetes anteriores y sus respectivos ACKs.*

## P2

- i) IP también ha sido implementado en líneas seriales como las telefónicas (se conoce como PPP). Discuta las ventajas y desventajas de usar TCP/IP en una línea de este tipo contra la alternativa de usar un protocolo ad-hoc de verificación de errores de nivel datos y nada más.

*Ventajas TCP/IP:*

- *multiplexión: puedo tener varias conexiones independientes que pasan por la misma línea serial*
- *no es necesario implementar nada nuevo y no necesito que la línea me garantice estar libre de errores*
- *es posible usar la red global sin necesitar nada adicional.*

*Ventajas protocolo ad-hoc:*

- *se pueden aprovechar características específicas de la línea para se-  
tear timeouts y demás criterios más adecuados que los de TCP/IP, que  
son genéricos.*
- *no se envía el overhead de los headers IP y TCP.*

- ii) En la capa datos, un paquete de datos lleva un header con un largo y un checksum. Luego se envía en varios frames físicos si excede el máximo tamaño.

Dado que no se usa un esquema de un paquete de datos en un frame físico, ¿para qué me sirven los frames? ¿no podríamos eliminarlos?

*Sin frames no sé donde buscar un comienzo de paquete si se me pierden bytes. Aunque vaya fragmentado en varios paquetes, sé que el comienzo del header viene en un comienzo de paquete, por lo que lo busco siempre después de un SOH.*

- iii) En la tarea 1 se deben implementar varios métodos de control de error. Según su punto de vista, el orden (little endian o big endian) influye en esa implementación? ¿Por qué?

*Depende de la implementación. Si se implementa procesando byte a byte, no habrá problemas. Pero si se usan tipos que abarcan más de un byte en lectura y/o cálculos intermedios, es necesario considerar el orden de los bytes, para que el cálculo de checksum en una máquina pueda repetirse en otra de distinta arquitectura.*

### P3

- i) Se tiene un layer de encapsulamiento que usa el siguiente formato:

SOH | type | DATA ..... | CC | EOP |

Cuando los datos a enviar son 1 byte, se obtiene un overhead de 4 veces el tamaño de los datos. Proponga un esquema especial para los paquetes de tamaño 1 que disminuya lo más posible el número de bits enviados, conservando la seguridad y sólo modificando el encapsulamiento de los paquetes de largo 1. No se pueden modificar los layers superiores.

Describe las modificaciones que se deben hacer al Frame layer visto en clases para implementar este esquema. ¿Es necesario modificar tanto el receptor como el emisor, o basta con modificar solamente uno de los dos?

*En la parte de la escritura de datos, se debe revisar si es de largo uno. Si es así, se encola en un buffer nuevo (llamémosle buf\_unos). Si este buffer sobrepasa un cierto número (por ejemplo MAX\_PACK-n), se envían todos esos bytes en un solo frame y se libera el buffer.*

*Si se quieren enviar datos de largo > 1, primero se revisa el buffer buf\_unos. Si tiene datos, primero se envían estos y luego se envía otro frame con los datos a enviar.*

*No es necesario modificar al lector, puesto que no es necesario que se lea la misma cantidad de bytes que se escribe. Si faltan bytes para completar lo que se quiere leer, se leen más frames hasta completar la cantidad necesaria. Si se quieren leer menos bytes de los que hay en un frame, la implementación considera guardar los restantes para una siguiente lectura.*

- ii) Hemos visto que se requiere que el tamaño máximo de la ventana sea uno menos que los números de secuencia disponibles. Muestre un ejemplo en que el algoritmo falle cuando ambos valores son iguales.

*Envío toda la ventana y espero los ACKs. Al otro lado se reciben bien y se envían todos los ACKs pero se pierden todos. Retransmito, y entonces al otro lado los recibo todos como válidos (nuevos), cuando en realidad son duplicados.*

## **PX**

Esta pregunta es opcional, pero permite mejorar el puntaje total de la prueba.

- i) ¿Puede un computador tener dos o más IP?

*Si, puede tener incluso más de una IP en una sola interfaz de red y ARP sigue funcionando sin problemas.*

- ii) ¿Pueden dos computadores tener las misma IP?

*(a) No, porque ya no sería posible diferenciar los dos computadores en la red.*

*(b) Si, siempre y cuando los dos computadores formen parte de dos redes que no tienen conexión entre ellas.*

- iii) ¿Qué diferencia a las IP privadas de las IP públicas?

*Las IP privadas no deben ser publicadas en tablas de rutas excepto las internas. Es posible que existan múltiples computadores que tengan la misma IP privada, pero una IP pública la puede tener un sólo computador.*

iv) ¿Qué ventaja tiene FDDI sobre Token Ring, y desventaja?

*FDDI provee redundancia, pero Token Ring tiene un mejor rendimiento y mucho menor costo.*

## P4

Considere que se hace un ping.

- i) Desde 146.83.4.9 a 146.83.4.11
- ii) Desde 146.83.4.11 a 209.88.207.132
- iii) Desde 146.83.4.9 a 146.83.1.1
- iv) Desde 146.83.4.33 a 200.27.2.2
- v) Desde 146.83.4.33 a 200.88.99.1
- vi) Desde 200.27.2.2 a 209.88.207.131

Detalle el camino de ida y vuelta de cada paquete, especificando la entrada en la tabla de rutas que se usa cada vez. El tamaño del ping es suficientemente pequeño para que no haya fragmentación.

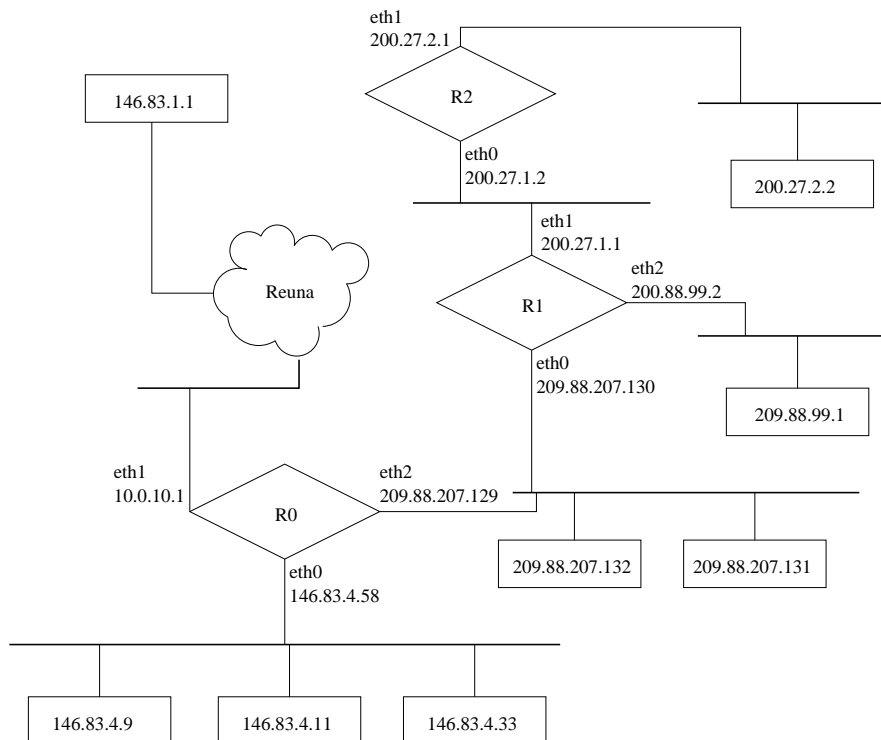


Figura 1: Topología de la red

| <b>Destination</b>                     | <b>Netmask</b>  | <b>Gateway</b> | <b>Interface</b> |
|--|-----------------|----------------|------------------|
| Tabla de ruta para 146.83.4.9, 11 y 33 |                 |                |                  |
| 146.83.4.0                             | 255.255.255.128 | 0.0.0.0        | eth0             |
| default                                | 0.0.0.0         | 146.83.4.58    | eth0             |
| Tabla de ruta para 209.88.207.132      |                 |                |                  |
| 209.88.207.128                         | 255.255.255.240 | 0.0.0.0        | eth0             |
| default                                | 0.0.0.0         | 0.0.0.0        | eth0             |
| Tabla de ruta para R0                  |                 |                |                  |
| 146.83.4.0                             | 255.255.255.128 | 146.83.4.58    | eth0             |
| 209.88.207.128                         | 255.255.255.240 | 209.88.207.129 | eth2             |
| 200.88.99.0                            | 255.255.255.224 | 209.88.207.130 | eth2             |
| 200.27.0.0                             | 255.255.0.0     | 209.88.207.130 | eth2             |
| default                                | 0.0.0.0         | 10.0.10.1      | eth1             |
| Tabla de ruta para 146.83.1.1          |                 |                |                  |
| 146.83.1.0                             | 255.255.255.0   | 0.0.0.0        | eth0             |
| default                                | 0.0.0.0         | 146.83.1.2     | eth0             |
| Tabla de ruta para 209.88.207.131      |                 |                |                  |
| 209.88.207.128                         | 255.255.255.240 | 0.0.0.0        | eth0             |
| default                                | 0.0.0.0         | 209.88.207.129 | eth0             |
| Tabla de ruta para R1                  |                 |                |                  |
| 209.88.207.128                         | 255.255.255.240 | 200.88.207.130 | eth0             |
| 200.88.99.0                            | 255.255.255.224 | 209.88.207.129 | eth0             |
| 200.27.0.0                             | 255.255.0.0     | 200.27.1.2     | eth1             |
| 200.88.99.0                            | 255.255.255.224 | 200.88.99.2    | eth2             |
| Tabla de ruta para R2                  |                 |                |                  |
| 200.27.2.0                             | 255.255.255.196 | 200.27.2.1     | eth1             |
| default                                | 0.0.0.0         | 200.27.1.1     | eth0             |
| Tabla de ruta para 200.27.2.2          |                 |                |                  |
| 200.27.2.0                             | 255.255.255.196 | 0.0.0.0        | eth0             |
| default                                | 0.0.0.0         | 200.27.2.1     | eth0             |