Diferencias de implementación y rendimiento en protocolos de transferencia confiable
Redes de Computadores I

Roberto Catricura
Loreto Godoy
Maxime Pawlak

6 de agosto de 2012

Índice

1. Resumen

Los protocolos de transporte confiable son de vital importacia en la transferencia de información entre redes de computadores y son apliamente usados en Internet.

En este trabajo acercaremos de manera practica a un tema visto en este curso, los protocolos de transferencia confiable, la idea es implementar los protocolos Alternating-Bit y Go-Back-N a través de un simulador el cual fue diseñado en lenguaje C [1], para hacer un envio de datos unidireccional a través de 2 nodos, con un output diseñado para poder ser analizado a simple vista y explorar su funcionalidad más a fondo, lo cual permite obtener conclusiones acerca de su rendimiento y difultad de implementación de manera practica.

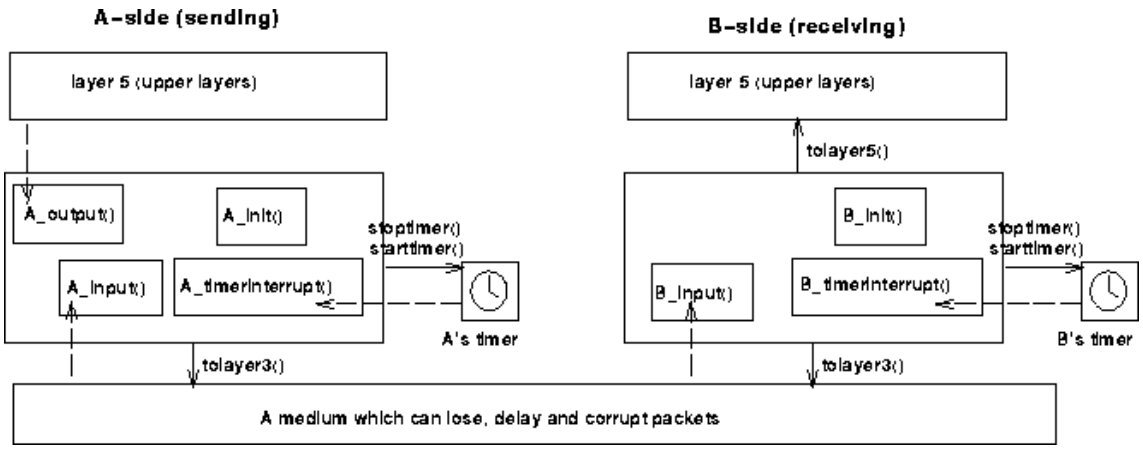
Se comienza desarrollando el protocolo más simple el cual corresponde a *Alternating-Bit*, nombre que recibe por los numeros de secuencia que usa en sus paquetes, alternandose entre '0' y '1' este funciona con una arquitectura de Stop-and-Wait, por lo que solamente enviara paquetes cuando recibe la confirmación de la llegada correcta del paquete anterior, es importante implementar de manera correcta estas funcionalidades ya que luego se volveran a utilizar, como lo es el checksum del paquete, se usara la llamada al sistema en Unix 'cksum' [2], por otro lado el timer utilizado en el protocolo viene implementado con el simulador, el cual ocupa funciones estandar de C.

Luego el protocolo *GBN* contiene funcionalidades más complejas, en primer lugar, se tiene el uso de una ventana que se desplaza para hacer el uso del *pipelining* del protocolo, es decir se define un cierto rango de paquetes que pueden ser enviados al mismo tiempo, se necesitará un buffer, para guardar una copia de los paquetes en caso de que ocurran errores en su transmisión, otra diferencia fundamental es el uso de *acknowledgements* o confirmaciónes acumulativas para los paquetes. Y por ultimo el uso de numeros de secuencia crecientes.

Como se dijo anteriormente, *GBN* tiene funcionalidades mucho más complejas que *Alternating-Bit* esto quedo demostrado, en la implementación. Si se habla de rendimiento se aprecia que *Alternating-Bit* es claramente menos eficiente refiriendose al tiempo necesario para enviar la misma cantidad de paquetes, gracias al *pipelining*, *Go-Back-N* es capas de transmitir una mayor cantidad de paquetes y confirmarlos en menor cantidad de tiempo, siempre y cuando no ocurra una corrupción o perdida constante de paquetes, ya que esto requerira enviar paquetes, nuevamente hacia el nodo receptor. Así concluimos el analisis de ambos protocolos de manera practica, obteniendo resultados similares a la teoria expuesta en el ramo, solo que de manera más practica y profunda.

2. Investigación

Al comenzar la implementación de ambos protocolos se investigo más a fondo sus funcionalidades ya que a diferencia de la parte teorica, este debe ser implementado de manera completa o al menos sus funciones más importantes, para esto se utilizo el libro guía [1] en su capitulo de 'Capa de Transporte' en su sección donde habla de los protocolos confiables, aquí se utilizan diagramas y *MEF* junto con psedocodigo para explicar paso a paso su implementación.
El siguiente diagrama corresponde a la implementación de los protocolos:



Se aprecia claramente el funcionamiento del simulador y cada uno de los procedimientos a implementar, estos son: *A_output, A_init, A_input, A_timerinterrupt, B_output, B_init, B_input y B_timerinterrupt*

Estas simulan el funcionamiento de una red entre 2 nodos, tambien se definen los siguientes procedimientos para el funcionamiento del simulador: *tolayer3, tolayer5, stoptimer, startimer.*

3. Implementación

Como se menciono anteriormente, la implementación comenzo por el protocolo más simple, correspondiente a *Alternating-Bit* este envia paquetes uno por uno, por lo que se implementa una iteración con un *flag* para reconocer el estado en que se encuentra el transmisor.

Luego se utiliza una estructura que define al mensaje y paquete en cuestión.

```
struct pkt {
   int seqnum;
   int acknum;
   int checksum;
   char payload[20];
    };
```

Aqui se puede ver que necesitaremos una implementación para definir el numero de secuencia de cada paquete (0 o 1), para el campo acknum se uso 0 para definir un *acknowlegment* negativo y 1 para el positivo, el checksum se realiza con una función de Unix [2] y el campo payload recibe el mensaje de la estructura correspondiente.
Existe también un respaldo de el paquete que se envia en caso de ocurrir errores, se definen condiciones en el receptor para confirmar la integridad del paquete y retornar una respuesta, la cual también pasa por un filtro para confirmar que no tenga errores.

En el caso del protocolo Go-Back-N se utilizan funcionalidades similares a las mencionadas anteriormente (no se explicara todo, por la limitación del informe) pero existen diferencias fundamentales, en GBN utilizamos un arreglo unidireccional para almacenar los paquetes que se envian, en caso de tener que reenviarlos, la ventana se maneja utilizando contadores, los cuales a traves de iteraciones van desplazando la ventana permitiendo la utilización de mayores numeros de secuencia, es clave también la utilización de confirmaciones acumulativas de los paquetes recibidos correctamente, esto se realiza leyendo la información del campo *seqnum* y comparandolo la variable *base* la cual se utiliza para saber el ultimo paquete confirmado de manera exitosa.

Finalmente la interrupción del *timer* que ocurre al perderse los paquetes, es similar en ambos protocolos, ya que esta esta definida por el simulador, no se necesita implementar este en si, si no la consecuencia de su detención. Por lo que se trata simplemente de volver a enviar los paquetes que no se han confirmado en el emisor.

4. Conclusiones

Con la implementación funcional de ambos protocolos se pueden realizar pruebas, para analizar su rendimiento, a manera de comprobar la teoria que se realizo en clases, se hacen 2 pruebas, con las siguientes parametros:

```
Caso 1:
Enter the number of messages to simulate: 25
Enter  packet loss probability [enter 0.0 for no loss]:0.0
```

```
Enter packet corruption probability [0.0 for no corruption]:0.0
Enter average time between messages from sender's layer5 [ > 0.0]:7
Enter TRACE:2

Caso 2:
Enter the number of messages to simulate: 25
Enter  packet loss probability [enter 0.0 for no loss]:0.1
Enter packet corruption probability [0.0 for no corruption]:0.1
Enter average time between messages from sender's layer5 [ > 0.0]:7
Enter TRACE:2
```

Con esta simple prueba podemos ver las diferencias de manera practica, con una corrupción y packet loss de 0 en los protocolos se obtiene para *Alternating-Bit* 9 mensajes transmitidos correctamente en 140[ms] mientras que para el protocolo *Go-Back-N* para 140[ms] se obtiene 15 mensajes, la diferencia es clara en rendimiento, cuando no existen errores ni perdida.

Ahora analizamos para el Caso 2, con corrupción y perdida de 0.1 obtenemos que en 140[ms] Alternating-Bit es capaz de transmitir 5 mensajes. Y Go-Back-N en 140[ms] transmite 10 mensajes.

Se ve claramente que al introducir errores en la transmisión de paquetes se compromete de manera grave el rendimiento de ambos protocolos, es de esperarse que al realizar pruebas con mayor error el protocolo GBN vea comprometida la cantidad de mensajes confirmados.

Referencias

[1] James F. Kurose, Keith W. Ross, *Computer Networking: A Top-Down Approach*

[2] `http://www.gnu.org/software/coreutils/manual/html_node/cksum-invocation.html`