Universidad de Chile
Facultad de Cs. Físicas y Matemáticas
Departamento de Ciencias de la Computación
CC4303 Redes

# Tarea 2 - Un protocolo de transporte apurete

Profesor: Catalina Álvarez

Auxiliar: Ivana Bachmann

## Objetivos

En esta tarea deberán construir una capa de transporte confiable sobre UDP, preocupándose de establecer la conexión, enviar paquetes de manera confiable y luego cerrar la conexión. Eso sí, su implementación será un poco distinta al estándar: deberán implementar Go-Back-N, pero van experimentar formas de abrir y cerrar la conexión.

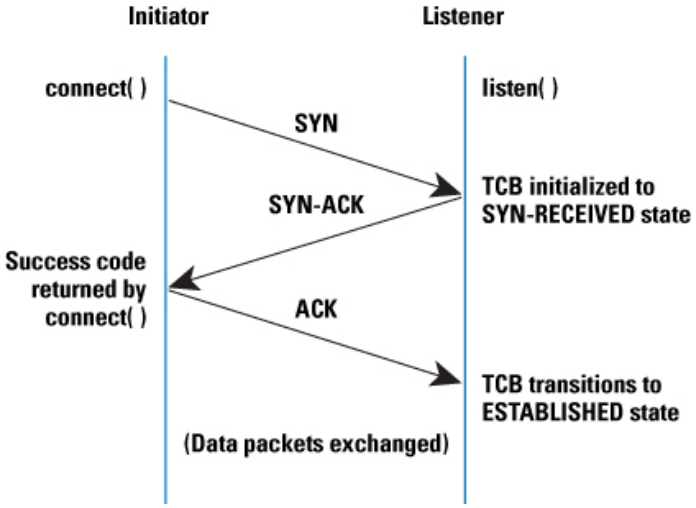
## Implementación pedida

Vamos a listar las características más importantes de lo que deben implementar:

- Un cliente y un servidor que implementen Go-Back-N. Su implementación debe soportar funcionar con delay, desorden y/o pérdida de paquetes.
  - El cliente lee un archivo de texto (cuya ruta se debe dejar como parámetro, ya sea en un archivo de configuración o pasado como argumento al llamar al cliente), y lo envía (por pedazos) al servidor.
  - Cada pedazo que envía el cliente debe estar empaquetado incluyendo un header. Este header debe incluir el número de secuencia del paquete y una flag indicando si el paquete es un ACK o lleva datos.
  - El servidor debe recibir los paquetes en orden, desempaquetarlos y escribirlos en un archivo de salida. Al finalizar la conexión, se espera que el archivo de entrada (el que lee el cliente) y el de salida (el que escribe el servidor) sean iguales.
  - El comportamiento del protocolo corresponde al clásico Go-Back-N: se manda una ventana entera y se esperan ACKs acumulativos. Si no se recibe la confirmación dentro de un tiempo determinado (timeout), se reenvía la ventana entera. Para distinguir los paquetes originales de sus retransmisiones, se utilizan los números de secuencia.

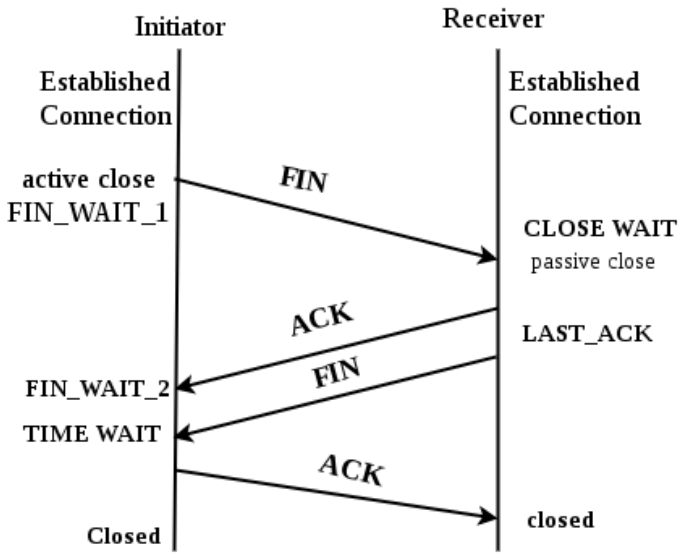
- El tamaño de la ventana es fijo, y ustedes tienen la libertad de definirlo (no puede ser 1 o estarían trabajando con un Stop-And-Wait).
- Para calcular los timeouts, deben usar el algoritmo de Karn.

- Cliente y servidor deben iniciar conexión antes de empezar con el envío de la información. En esta tarea tienen que implementar dos formas de inicio de conexión:
  - Three-way hanshake: la forma estándar, que incluye mandar un paquete de conexión inicial, y dos otros para sincronizar. Se entiende más fácilmente en la figura siguiente.




  - Two-way hanshake: igual que el anterior, pero el cliente no confirma la llegada del paquete SYN-ACK.

- Cliente y servidor también tienen que cerrar la conexión correctamente, como se muestra en la imagen siguiente. Es importante que tengan cuidado de que tanto cliente como servidor terminen la conexión correctamente.




- Finalmente, deben considerar el caso en que cliente o servidor se caen, o bien se haya perdido la conexión; para estas situaciones, se les recomienda incluir un mecanismo de máximas retransmisiones, y terminen los procesos en caso de que se llegue a este número (esto puede ocurrir intentando enviar data o ACKs).

### Pruebas de eficiencia

Una vez terminada la tarea, se les pide probar su implementación (utilizando ambas formas de conexión) con pérdida de 0, 10 % y 30 %, y con delays de 0, 100ms y 500ms; además, deben incluir pruebas en que aleatoriamente cliente y/o servidor se caigan inesperadamente, y casos en que la conexión se pierda (emulable con pérdida del 100 %). No es necesario que prueben todas las combinaciones posibles, pero sí que sean exigentes con su tarea (se recomienda al menos probar el valor máximo de pérdida con 100ms).

Para realizar las pruebas, pueden usar "netem" en Unix:

```
% tc qdisc add dev lo root netem loss 20.0% delay 100ms
```

Y tienen 20 % de pérdida con 100 milisegundos de delay. Para modificar el valor, deben usar **change** en vez de add en ese mismo comando.

### Informe

Igual que para la tarea anterior **deben entregar un README explicando como correr su tarea**, y aclarando sus supuestos. Por favor consideren que mientras mejor documentada su tarea, más fácil su evaluación y menos problemas tendrán.

Además, deberán entregar un pequeño informe con:

- Los resultados de las pruebas que realicen. Deben incluir el tamaño del archivo que enviaron, el tiempo que toma el envío, el número de retransmisiones y si acaso la conexión falló o no.
- Análisis de los resultados: ¿En qué condiciones falla más/menos el protocolo? ¿Qué podrían optimizar? ¿Es mejor three o two handshake? ¿Porqué?

### Implementación

Ustedes deben implementar lo pedido en uno de los siguientes lenguajes de programación: **Java**, **C**, **C++** o **Python**. En cualquiera de los casos, deben consultar al auxiliar o profesor en caso de que deseen usar alguna librería/framework fuera del core del lenguaje.

La posibilidad de utilizar otros lenguajes no está cerrada, pero debe conversarse con la auxiliar previamente (básicamente para asegurarse de que su tarea sea revisable).

Cualquier duda o pregunta o reporte de bugs, dirigirse al foro de U-cursos.

### Evaluación

Esta tarea será evaluada tomando en cuenta la funcionalidad de su cliente y servidor, el análisis entregado en el informe y la facilidad/dificultad que tenga el corrector de hacerla funcionar.