TP D: Desarrollo de aplicaciones distribuidas con BSD sockets

Fecha y hora límite de entrega: 23/11 23:59:59

Carácter: grupal (24 grupos de 3 y 1 grupo de 2).

Deben enviar por correo y con asunto grupo TP D los integrantes del grupo y a su vez éstos deben estar copiados en el mail.

Objetivo

Que los alumnos se familiaricen con la biblioteca BSD Sockets y se interioricen con las particularidades de TCP y UDP. Que logren intercomunicar sistemas y sepan reconocer en qué casos conviene utilizar cada capa de transporte. El trabajo práctico consta de dos partes.

En la primera parte, los alumnos deberán desarrollar un protocolo de transferencia de archivos basado en *stop & wait* utilizando UDP. Deberán implementar el cliente y el servidor. Deberán validar su correcto funcionamiento contra un servidor alojado en Internet por la cátedra. Deberán hacer pruebas en 3 escenarios distintos y extraer conclusiones de su performance, plasmándolo en respuesta a las preguntas (ver Entregables).

En la segunda parte, deberán desarrollar un aplicación cliente servidor en TCP para medir el one-way delay y analizar el comportamiento de TCP como servicio de transporte bajo condiciones de red desfavorables (packet loss, jitter). Los estudiantes deberán implementar el *framing* debido a que TCP es *byte stream oriented* y el tamaño de las PDU será variable.

Primer parte: UDP stop & wait file transfer protocol

Descripción de la aplicación

La aplicación se utilizará para enviar archivos desde el cliente a hacia el servidor. Para acotar el alcance solo se solicitará implementar la subida de archivos (Write Request). Se debe respetar la especificación protocolar dada a continuación.

Especificación

- El procedimiento de transferencia de un archivo consta 4 pasos
    - Autenticación (HELLO)
    - Parametrización (WRITE REQUEST)
    - Transferencia de datos (DATA)
    - Finalizacion de la sesión (FINALIZE)
- El protocolo utilizará el puerto 20252 en el lado del servidor y un puerto efímero en el cliente
- La PDU encapsulada en UDP tiene el siguiente Formato.

```
+---------------+----------------+----------------------+
|    Type (1B)   | Seq Number (1B) |  Data (optional)      |
+---------------+----------------+----------------------+
```

- Los posibles valores de Type son: HELLO(1), WRQ(2), DATA(3), ACK(4), FIN(5)
- Los posibles valores de Sequence Number son 0 y 1 dado que es un protocolo stop & wait
- El servidor debe soportar concurrencia de hasta N clientes (N es un parámetro definido en tiempo de compilación). Pista: implica llevar un estado de cada cliente en una estructura.
- El valor máximo de Data podría ser el de máximo permitido por IP (aunque fragmente) aunque se recomienda un valor de 1478=1500-20-2.

Detalle de la interacción protocolar

Los errores que no estén explicitados en las siguientes fases deberán ser descartados silenciosamente, es decir el servidor no informará al cliente. Sin embargo, es altamente recomendable que se informen en pantalla o se registren en un log file.

Fase 1: autenticación (HELLO)

Step	Sender	Receiver	Payload	Description
1	Client	Server	**Credentials**	El cliente envía una PDU de tipo HELLO(1) con Seq. Num. 0 y el payload contendrá la credencial entregada por el docente a cada grupo de alumnos.
2	Server	Client	Empty or Error message	Server valida las credenciales. Si son válidas envía una PDU ACK(4) con Seq. Num. 0. Si son Inválidas responde con un ACK(4) con Seq. Num. 0 pero con un Payload indicando el mensaje de error para imprimirlo al usuario en pantalla

Fase 2: Parametrización de la operación (WRITE REQUEST)

Step	Sender	Receiver	Payload	Description
3	Client	Server	**Filename**	El cliente envía una PDU de tipo WRQ(2) con Seq. Num. 1 y el payload contendrá el nombre de archivo en el servidor (NULL terminated string). Sólo se aceptan caracteres ASCII. El Filename debe tener como mucho 10 caracteres y como mínimo 4 caracteres. Se proveerá de un archivo a cada grupo para la validación contra el servidor de la cátedra.
4	Server	Client	Empty or Error message	Server valida el Filename. Si es válido envía una PDU ACK(4) con Seq. Num. 1. Si es inválido responde con ACK(4) con Seq. Num. 1 pero con un Payload indicando el mensaje de error para imprimirlo al usuario

Fase 3: Transferencia de datos (DATA)

Step	Sender	Receiver	Payload	Description
5	Client	Server	**File Data**	El cliente envía el primer trozo de información en una PDU de tipo DATA(3) con Seq. Num. 0 e inicia el timer de retransmisión.
6	Server	Client	Empty	El servidor recibe los datos, escribe en el archivo y envía una PDU ACK(4) con Seq. Num. 0
7	Client	Server	**File Data**	El cliente incrementa el Seq. Num. Envía el siguiente trozo de información en una PDU de tipo DATA(3) con Seq. Num. 1 e inicia el timer de retransmisión.
8	Server	Client	Empty	El servidor recibe los datos, escribe en el archivo y envía una PDU ACK(4) con Seq. Num. 1
…				Se repiten los pasos 5 al 8, donde se van alternando los Seq Num (0, 1, 0, 1, …)

Fase 4: Finalización (FIN)

Step	Sender	Receiver	Payload	Description
N	Client	Server	**Filename**	Después de recibir el último ACK, el cliente incrementa el Seq. Num. y envía una PDU de tipo FIN(5) con el Seq. Num. que corresponda.
N+1	Server	Client	Empty or Error message	Server valida el Filename. Si es válido envía una PDU ACK(4) con el Seq. Num que corresponda. El Servidor libera los recursos.

Consideraciones a tener en cuenta

- El Seq. Num. debe ser incremental, si antes fue 0, el próximo debe ser 1 si no hay retransmisión. El servidor registrará eventos de este tipo.
- No se puede enviar el próximo PDU de datos antes de recibir su correspondiente ACK (el servidor registrará eventos de este tipo)
- Se debe respetar estrictamente las 4 fases. Ejemplos de incumplimiento:

    - Si se envía un WRITE REQUEST sin su correspondiente paso de autenticación se debe descartar silenciosamente
    - Si se envía una PDU de DATA sin antes haber realizado un WRITE REQUEST se debe descartar silenciosamente

Pruebas a superar

- La cátedra proveerá un archivo a cada grupo y verificará el digesto del archivo subido. Esto implica que se hagan las 4 fases en el orden correcto.
- El servidor implementado por la cátedra podrá omitir intencionalmente el envío de ACK (de esta manera medirá funcionamiento y el tiempo de retransmisión)
- El servidor implementado por la cátedra podrá demorar intencionalmente los ACK para probar el control de flujo del protocolo stop & wait
- El servidor implementado por la cátedra podrá responder con ACK con Seq Num incorrecto que deberán ser ignorados por el timer del cliente.

Segunda parte: TCP performance

Descripción de la aplicación

El cliente se conecta a un servidor y envía PDU a intervalos de tiempo regulares. Para poder medir el mejor ello el cliente y servidor deben estar sincronizados (se recomiendan sincronizar las PCs / Virtual Machines por NTP (googlear: *how to synchronize time using NTP in ubuntu 24*)

Especificación

- La Protocol Data Unit debe tener la siguiente estructura

Field	Size (Bytes)	Description
Origin Timestamp	8	high-resolution timestamp recorded just before transmission (la unidad es microsegundos desde el 1/1/1970 UTC)
Payload	N	A variable number of filler bytes $500 <= N <= 1000. All bytes must be 0x20
Delimiter	1	The final byte, which must be the ASCII character 124

  El tamaño total de la PDU puede variar desde 8+500+1 = 509 bytes hasta 8+1000+1=1009 bytes.
- El server utiliza el port 20252.
- La unidad del Origin Timestamp son microsegundos, por ende se recomienda usar `gettimeofday()`. El tipo de datos uint64 (unsigned 64-bit integer) soporta un valor máximo de 18,446,744,073,709,551,615 que es muy superior a los 16 dígitos necesarios para expresar la fecha epoch en microsegundos.
- Para simplificar el server, no es necesario que este sea capaz de atender varios clientes concurrentemente.

- Para simplificar se va a suponer igual endianness entre cliente y servidor (sino habría que usar las funciones `htobe64/be64toh`)
- El server debe ser capaz de soportar lecturas parciales (justamente es la característica que surge de que TCP es byte stream oriented y las PDUs son de largo variable), Podría ocurrir que en un solo `read()` se lean entre 0 y 2 PDUs **completas** por ejemplo.
- Una vez que la PDU es recibida **en forma completa** se debe extraer el Origin Timestamp y luego calcular el One-Way Delay (Delay = Destination Timestamp - Origin Timestamp). Es considerado incorrecto, obtener el Destination Timestamp si todavía no se extrajo la PDU en forma completa (es decir, se vió el delimitador "|").
- Se debe loguear este one way delay, en un archivo CSV. Las columnas serán: `Número de medición, Valor One-Way Delay`. El one-way delay está medido en segundos con alta precisión. Por ejemplo:
  ```
  1, 0.10002
  2, 0.10001
  3, 0.10350
  ```
- El cliente debe tener los siguientes parámetros desde consola:
  - tiempo entre timestamp d milisegundos (típicamente usando `usleep()`)
  - Duración total de la prueba de envío: N segundos

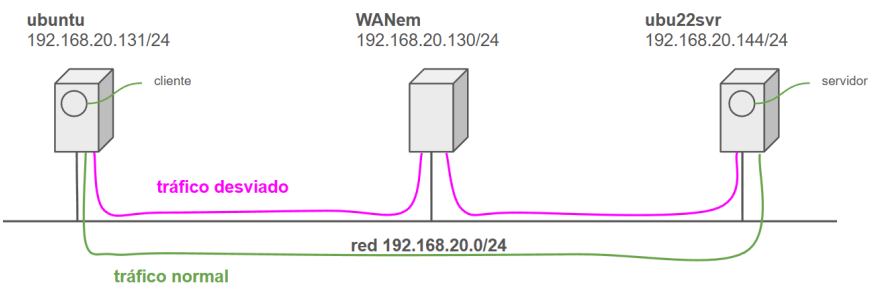
  Ejemplo: `./cliente -d 50 -N 10`

Pruebas a superar

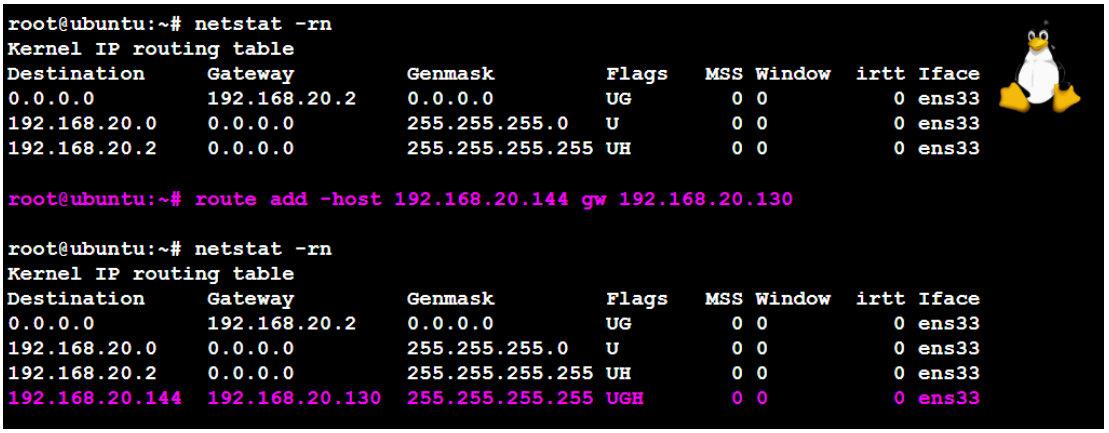
- Deberán probar cliente y servidor en un ambiente de simulación tipo NetEm (Linux). Se recomienda usar WANem (https://wanem.sourceforge.net/) para no tener que conocer la sintaxis del utilitario `tc` (traffic control) que es bastante compleja
- La maqueta es similar a la utilizada en el workshop de TCP/UDP. Recuerde que para ello se deben cargar rutas de hosts en las VMs. Tanto en la que contiene al cliente como en la que contiene al servidor.

Emulador de WAN: WANem setup

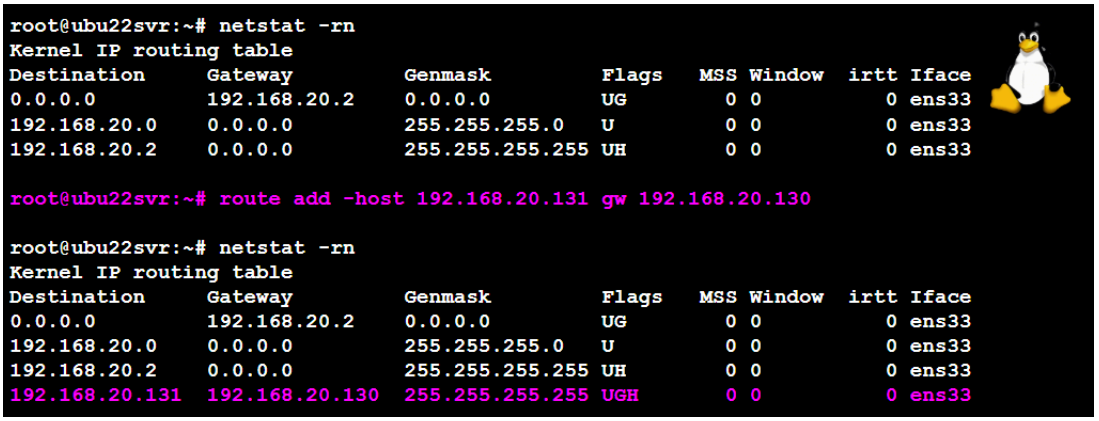
- La maqueta consta de 3 VMs en la misma red LAN



En la VM ubuntu se agrega con el comando route la ruta de host y se fuerza a que pase por el intermediate system que tiene la IP del WANem



En la VM ubu22svr también se agrega la ruta de host correspondiente



- **[Entregable]** Se deberá simular un ambiente de pérdidas del 1%, 2%, 5%. Se deberá graficar el one-way delay en función del número de medición para cada caso.
- **[Entregable]** Se deberá simular un ambiente de jitter con un delay de 50 ms y jitter de 40 ms. Se deberá graficar el one-way delay en función del número de medición.

Preguntas a contestar

1. Transfiera un archivo de 20 kB en los siguientes 3 escenarios:
   (generelo con el comando: `dd if=/dev/zero of=~/archivo_20kB bs=1024 count=20`)
   Escenario 1 **LAN**: LAN virtual de las virtual machines
   Escenario 2 **Latencia internacional**: emulación de delay de 90 ms en cada sentido (RTT ~180 ms)
   Escenario 3 **Latencia a una base lunar**: emulación de delay de 1,28 s en cada sentido (RTT ~2,56s)
   ¿Cuánto tarda en cada caso? **mostrar evidencias en wireshark (capture en el cliente)**
   Deduzca la fórmula de valor teórico de transferencia (tenga en cuenta de contemplar las 4 fases) y compararlo contra lo evidenciado en Wireshark.
2. ¿Recomendaría este protocolo para actualizar un firmware de 16 MB de un equipo remoto en Virginia(EEUU) desde Bs As?
   Verificar la latencia a Virginia usando https://www.cloudping.info/

3. ¿Qué conclusión extrae sobre TCP en ambiente de elevado packet loss para uso en aplicaciones de tiempo real? ¿Qué ambiente que utiliza habitualmente podría contener elevado packet loss o jitter, por qué?

Recomendaciones generales

- En el caso de UDP con concurrencia es fundamental entender que no se puede optar por una estrategia multi-process ya que es el mismo socket el que recibe de todos los clientes a la vez (recordar demultiplexación en UDP del workshop de TCP/UDP). Es necesario controlar la disponibilidad del recurso socket para operaciones de lectura y escritura usando la system call `select()` o `poll()` en su defecto.
- Se sugiere loguear en pantalla y/o en un archivo las sesiones que el server establece para realizar las transferencias,
- Usar el comando `nc` para ir probando pequeños códigos (ejemplo: `nc -k -l 0.0.0.0 20252`)

Entregables

- Se deberá entregar el **código** junto con sus instrucciones de uso y compilación.
- Se deberá entregar un documento con las respuestas a las preguntas y otros entregables especificados a lo largo del enunciado.
- Se deberá hacer una breve presentación Powerpoint **grabada en video** orientada a sus pares (es requisito **obligatorio usar Youtube**) que será expuesta el 23/11/2025 y el 28/11/2025. Se debe mostrar brevemente la arquitectura diseñada y un diagrama de la misma. A su vez se debe hacer énfasis en la experiencia personal de programar usando BSD Sockets (desafíos, decisiones tomadas, dificultades, expectativas, etc). La presentación deberá durar entre **6** y **8 minutos.**
- También se debe **adjuntar el PDF de la presentación y el link al video.** La carátula de la presentación debe identificar claramente los integrantes del grupo.

Cabe la posibilidad de que se haga alguna pregunta a cualquiera de los integrantes del grupo al respecto de lo expuesto o relacionado al código o a la estrategia que se utilizó para satisfacer los requerimientos de las aplicaciones. Es obligatorio conocer el funcionamiento detallado del código, sino deberemos suponer desconocimiento por no haberse involucrado activamente en el desarrollo del Trabajo Práctico.

Ejemplos de videos:
https://www.youtube.com/watch?v=i8xFl0kYUEw
https://www.youtube.com/watch?v=MpisJFpIqIk
https://www.youtube.com/watch?v=34XIPz9ZWwQ