Control 1
Redes
Plazo de entrega: 9 de abril de 2025

José M. Piquer

Intro

El control se responde en forma individual, sin preguntas entre Uds o a terceros. Todas las dudas pregúntenlas en el foro de U-cursos y el equipo docente les responderá. Entreguen las tres preguntas en un PDF con su nombre por U-cursos. Pueden (y deben) buscar material en Internet para responder, pero citen siempre las fuentes que utilizan. Pueden usar chatGPT para ayudarlos, pero no pueden entregar su respuesta directamente, deben explicar y analizar todo lo que les propone. Recuerden que chatGPT se equivoca, alucina y miente con mucha convicción. Siempre recuerden mencionarlo si lo utilizan.

En épocas de chatGPT estamos obligados a evaluar mal respuestas que antes habríamos aceptado como casi correctas. Si Uds entregan un párrafo de argumentos correctos pero no responden nunca la pregunta específica con argumentos técnicos que corresponden a ese caso, hoy en día tienen cero puntos.

Un ejemplo, decir algo como:"existen algoritmos propuestos con mejoras para TCP para enlaces de alto delay como CUBIC y BBR, que buscan resolver este problema" antes les habrían dado algún puntaje (significaba que habían hecho una búsqueda en google al menos). Hoy en día vale cero.

Las preguntas de los controles las chequeamos con chatGPT antes de ponerlas, y la idea es que chatGPT no logre sacarse un 4.0.

Entonces, la recomendación es: contesten la pregunta específica del enunciado. Las frases periféricas que buscan adornar sin ir al detalle técnico no aportan en nada, de hecho más bien perjudican, por que mientras más hablan sin ir al punto, más generan la impresión en el corrector que no conocen la respuesta (y suenan igual que chatGPT).

P1: sockets

En IOT (Internet Of Things) es común tener sensores que se conectan a internet para enviar los valores obtenidos por el sensor.

En una plantación de tomates se utiliza un cálculo de energía absorbida en base a la temperatura registrada durante el día. Mediante este cálculo se decide cuándo cosechar los tomates. Para minimizar las pérdidas por maduración excesiva o insuficiente se han instalado sensores IOT por cada 100 m2 en un campo de 10 hectáreas (1 hectárea son 10.000 m2). En cada sector se instalan sensores de temperatura, humedad y radiación UV. Se considera que 2 mediciones al día por sonda son suficientes para calcular la cosecha. Se le solicita definir un protocolo entre las sondas y el servidor que recibe las distintas mediciones.

Ejemplo (≫ indica mensaje del servidor, ≪ indica mensaje de la sonda):
≫01 Hola, soy servidor v.01
≪e001 soy la sonda57345
≫02 Bienvenido sonda 57345
.
. texto borrado
.
≪fin

1. Discuta si usaría TCP o UDP
2. ¿El servidor debería conectarse a las sondas o las sondas al servidor?
3. ¿El servidor debería estructurarse con procesos pesados, threads o select?
4. Defina el protocolo para que el servidor reciba los datos

P2: protocolos

Conteste las siguientes preguntas:

1. Al momento de desarrollar una aplicación conectada a la red previo a la introducción del modelo OSI de capas, y en particular a sockets en Internet, existían varios problemas. Según su criterio, cuáles eran los 2 principales problemas que se resolvieron con OSI y sockets.

2. ¿Qué problemas existen al implementar protocolos binarios para la comunicación entre un cliente y un servidor? ¿Puede ser bueno un protocolo binario en algunos casos?

3. Como se vio en clases, cada día se utilizan más protocolos "seguros" como HTTPS. Un ingeniero dice que eso fue pésima idea, por que hubo que modificar el protocolo mismo para poder encriptar/desencriptar la información. ¿Tiene razón ese ingeniero?

4. Para la Tarea 1, tuvimos que modificar el cliente TCP `client_echo3.py` para medir el ancho de banda disponible. Las últimas líneas de ese cliente son:

   ```
   time.sleep(3) # dar tiempo para que vuelva la respuesta
   s.close()
   ```

   Para medir ancho de banda, obviamente es una mala idea esperar 3 segundos antes de terminar. Proponga formas de eliminar ese sleep() y sin embargo asegurar que todas las respuestas hayan llegado antes de cerrar el socket.

P3: UDP

Se le solicita que implemente un cliente que envíe paquetes UDP con el id del computador, id de la cpu, la carga CPU (es un número entre 0 a 100) y la fecha en formato aaaammddhhMMSS (ej 20250407160000) a un servidor cada M minutos. El servidor responderá con un ACK y la fecha, el id del computador y el id de la cpu cuando el paquete sea recibido, para almacenarlo en una base de datos. Si no se recibe el paquete, el cliente debe enviarlo cada S segundos por R intentos, luego descarta el paquete e intenta con la siguiente medición. Implemente el cliente y el servidor en python, teniendo en consideración:

- El id del computador, id de la cpu y la fecha son las llaves para la base de datos por lo que no se pueden repetir.
- El servidor puede recibir conexiones desde distintos clientes y un mismo cliente puede enviar distintos paquetes dependiendo del número de CPUs que tenga.
- Para la implementación pueden usar un archivo de texto donde el servidor va almacenando las líneas de texto recibidas, en vez de una base de datos de verdad.

P4: Proxy UDP

En el curso vimos cómo construir un proxy TCP, más o menos general. Ahora pensemos en qué es un proxy UDP. Lo difícil es que no hay conexiones propiamente tales, entonces no es tan fácil descubrir a qué cliente le corresponde una respuesta.

Le pedí a chatGPT que me programara un proxy UDP en python y me propuso el siguiente código[1]:

```
import jsockets
import threading

# Configuración del proxy
PROXY_PORT = 9999 # Puerto del proxy
TARGET_HOST = "anakena.dcc.uchile.cl" # Dirección del servidor de destino
TARGET_PORT = 1818 # Puerto del servidor de destino

# Diccionario para mapear clientes a servidores
client_map = {}

# Crear socket UDP
proxy_socket = jsockets.socket_udp_bind(PROXY_PORT)

def handle_client():
    while True:
        data, client_address = proxy_socket.recvfrom(1024*1024)

        if client_address not in client_map:
            client_map[client_address] = (TARGET_HOST, TARGET_PORT)

        target_address = client_map[client_address]

        # Reenviar el paquete al servidor de destino
        proxy_socket.sendto(data, target_address)

        # Recibir respuesta del servidor de destino
        response, _ = proxy_socket.recvfrom(1024*1024)
```

[1] Sólo cambié algunos detalles para hacerlo más parecido a los códigos del curso, como el uso de `jsockets` y esas cosas.

```
        # Enviar respuesta de vuelta al cliente
        proxy_socket.sendto(response, client_address)

# Iniciar el hilo para manejar clientes
threading.Thread(target=handle_client, daemon=True).start()

print(f"Proxy UDP escuchando en {PROXY_PORT}, redirigiendo a {TARGET_HOST}:{TARGET_PO

# Mantener el proxy en ejecución

try:
    while True:
        pass
except KeyboardInterrupt:
    print("Proxy UDP detenido.")
    proxy_socket.close()
```

Está bastante malo. Se les pide que lo corrijan para que funcione. No lo escriban entero de nuevo, solo modifiquen lo que encuentren necesario, manteniendo la idea original que tenía chatGPT (¡si se puede decir eso!). Pueden probarlo con el servidor de la T2, por ejemplo:

```
% ./proxy-udp-C1.py &
% ./client_echo2_udp.py localhost 9999
hola
hola
...
```

Tengan cuidado al probarlo, que tiende a quedarse en un ciclo infinito enviando/recibiendo paquetes con anakena, asegúrense de matarlo.