

### **Trabajo Práctico:** Sockets TCP/UDP

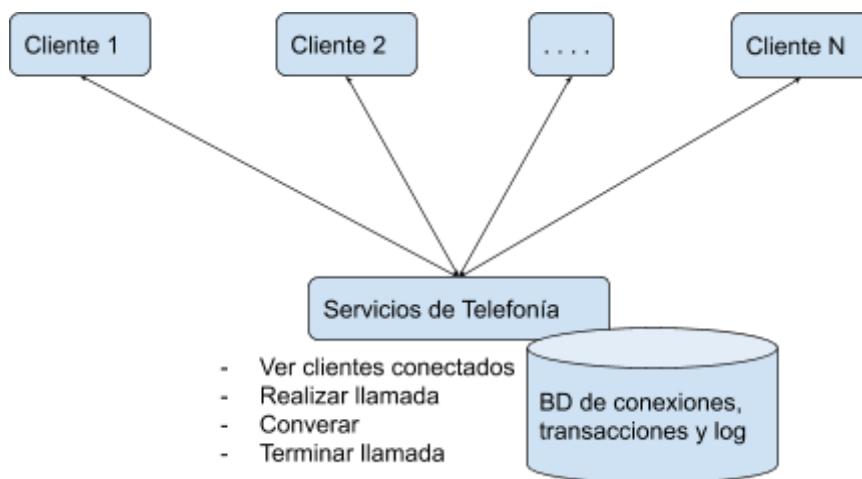
**Objetivo principal:** Realizar el diseño y desarrollo de sistemas distribuidos utilizando comunicación entre procesos sobre protocolos TCP y UDP y comprender las dificultades de su implementación.

#### **Objetivos específicos:**

- Enfrentarse a la problemática de diseñar sistemas que provean servicios e interfaces a otros sistemas.
- Comprender los desafíos en el proceso de desarrollo de integración de sistemas.
- Desarrollar e implementar sistemas basados en envío y recepción de paquetes utilizando protocolo UDP.
- Desarrollar e implementar sistemas utilizando flujos de comunicación utilizando protocolo TCP.

#### **Especificación del sistema distribuido**

El objetivo del trabajo es contar con aplicaciones distribuidas geográficamente basadas en Sockets TCP y UDP que permitan operar eficientemente a las organizaciones involucradas.



#### **Proyectos a desarrollar e implementar**

Cada grupo deberá crear y desarrollar los siguientes proyectos:

- Un proyecto para el Sistema de Servicios de mensajería y/o llamadas
  - Debe ofrecer servicios sobre TCP para las llamadas y servicios UDP para las demás operaciones.
  - Debe recibir peticiones para conectar a los clientes.
  - Debe tener conectado a todos los clientes que quieren realizar llamadas telefónicas entre sí.
  - Una llamada telefónica será representada como un intercambio de mensajes de texto bidireccional en tiempo real. En vez de enviar y recibir audio, se transmitirá texto con el mismo funcionamiento.

- Las conexiones de clientes activos se almacenan en memoria (Array).
- Para realizar una llamada o conversación entre 2 clientes conectados, deben encontrarse en el Array de clientes del Servidor. El servidor debe crear un hilo para manejar dicha situación.
- Operaciones que debe soportar:
  - Ver clientes conectados: Muestra a un cliente conectado al servidor la lista de todos los clientes que actualmente están. Se propone por el canal del protocolo UDP.
  - Realizar llamada. Procedimiento que intenta conectar a 2 clientes. Debe dar un error controlado en caso que el cliente destino esté ya ocupado.
  - Conversar. Debe permitir el envío y recepción de mensajes entre 2 clientes pasando por el servidor.
  - Terminar llamada. Debe liberar a los clientes a fin que cada uno pueda realizar otra llamada. Se propone por el canal del protocolo UDP.
- Adicionalmente el servidor deberá guardar un registro estadístico de las conexiones realizadas en un log.
  - El log puede ser un archivo, una estructura almacenada en memoria ó una tabla en base de datos.
  - Debe contener: fecha-hora, origen (ip:puerto), destino (ip:puerto).
- Un proyecto Cliente de Servicios de Telefonía
  - Debe ser una aplicación cliente consola (standalone) o con interfaz gráfica que simule una llamada (las comunicaciones se implementan en texto)
  - Debe implementar los servicios del Servidor de Telefonía.
  - Debe soportar lo siguiente:
    - Ver clientes conectados.
    - Realizar llamada. Conexión establecida con un cliente.
    - Conversar (enviar y recibir mensajes al cliente que se conectó).
    - Terminar llamada.

### **Representación de Datos**

Para la representación de datos entre debe utilizarse el formato/notación JSON. Cada interacción entre los intervenientes debe contener mínimamente los siguientes atributos:

- estado, donde:
  - "0" corresponde a una transacción exitosa.
  - "-1" transacción indeterminada.
  - Mayor a "1" un código de error o mensaje relacionado a la transacción.
- mensaje
  - Palabra "ok" si no existe error.
  - El detalle el error si existe.
- tipo\_operacion:
  - 1: ver clientes conectados

- 2: iniciar llamada
- 3: conversar. Se puede tener 2 códigos uno para envío y otro de recepción.
- 4: terminar llamada.
- Otros a considerar
- Cuerpo u otros datos específicos según el tipo de operación.

**Se debe:**

- Desarrollar aplicaciones clientes y servidores necesarios para el cumplimiento del sistema solicitado.
- Cada componente del sistema distribuido debe tener su propio proyecto de software, es decir: cada componente/sistema debe estar en carpetas separadas sin que estén relacionadas entre sí a nivel de código fuente. Esto a fin de cumplir con la independencia y autonomía de cada sistema.
- En relación al repositorio deben tener un solo proyecto GIT en plataforma gitlab.com o github.com. El repositorio debe ser visible para el profesor (usuario: fmancia). Dentro del repositorio debe haber una carpeta diferente para cada proyecto.
- Definir la estructura de datos que cada fuente de datos debe manejar. Utilizar JSON como representación de datos para el intercambio.
- Diseñar el mecanismo de prueba sencillo y simple para la demostración frente al profesor y audiencia.
- Diseñar una interfaz en el cliente acorde a las pruebas.
- Crear una nomenclatura y jerarquía ordenada para cada archivo de código fuente que necesiten. Será valorada dicha organización.
- Dentro del repositorio del proyecto debe contener un archivo "README.md" donde se especifique:
  - Los nombres de los alumnos que realizaron el trabajo.
  - Requerimientos de instalación.
  - Cómo crear la estructura de Base de datos.
  - Cómo poblar los datos iniciales necesarios de Base de datos.
  - Cómo compilar y ejecutar los componentes de cada servidor.
  - Cómo compilar y ejecutar el/los clientes.
  - Documentación de un API de servicios ofrecidos por el Servidor.
  - Especificar la forma de invocación y parámetros de cada servicio ofrecido por el servidor.
  - El API documentado debe ser leído e implementado por otros grupos de desarrolladores.
- No se permiten plagios.

**Entrega**

- Grupos de 4 o 5 integrantes.
- Fecha de entrega: 11/04/2020
- Forma de entrega: Uno de los integrantes debe enviar un email al profesor con el link al repositorio.