Universidad de los Andes
Facultad de Ingeniería

Departamento de Ingeniería de Sistemas y Computación
Infraestructura de Comunicaciones

ABET Engineering Accreditation Commission

Profesor Carlos Lozano
calozanog@uniandes.edu.co

Asist. de laboratorio: Nathalia Quiroga
n.quiroga@uniandes.edu.co

Laboratorio #3

ANÁLISIS CAPA DE TRANSPORTE Y SOCKETS

1. OBJETIVO (S)

Al finalizar la práctica el estudiante estará en la capacidad de:

- Comprender el funcionamiento de los sockets en C como mecanismo de comunicación entre procesos.
- Diseñar un modelo de publicación–suscripción que permita la distribución de mensajes entre múltiples clientes.
- Analizar y comparar el comportamiento de TCP y UDP en términos de confiabilidad, control de flujo, orden y pérdida de mensajes.
- Utilizar herramientas de captura de tráfico (Wireshark) para observar las diferencias entre ambos protocolos.

2. LECTURAS PREVIAS

- Kurose, J., & Ross, K. Computer Networking: A Top-Down Approach. Sección 2.7: “Sockets Programming”.

3. CONTEXTO

3.1 ¿Qué es un Socket?

Un socket es un punto de conexión de comunicación, a través del cual una aplicación envía o recibe paquetes de datos a través de una red.

Un socket se identifica de forma única por un par de valores:

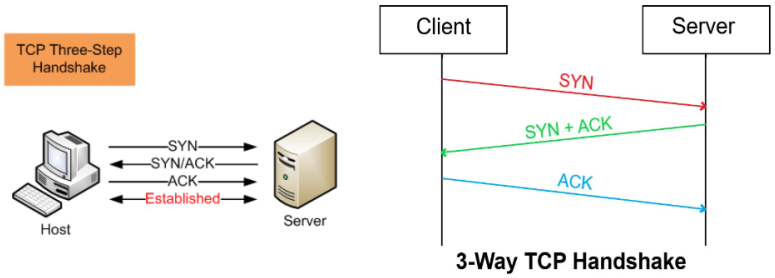
- **Dirección IP** – La dirección de la máquina que participa en la comunicación.
- **Puerto** – Un número que identifica el servicio o la aplicación específica en esa máquina.

Por ejemplo, un socket puede representarse como 192.168.1.10:8080.

3.2 TCP (Transmission Control Protocol)

TCP es un protocolo orientado a la conexión. Establece un canal confiable entre dos extremos mediante un handshake de tres vías. Sus principales características son:

- Confiabilidad: garantiza que los datos lleguen al destino.
- Orden: asegura que los mensajes lleguen en el mismo orden en que fueron enviados.
- Control de flujo y congestión: ajusta la velocidad de transmisión según el estado de la red.



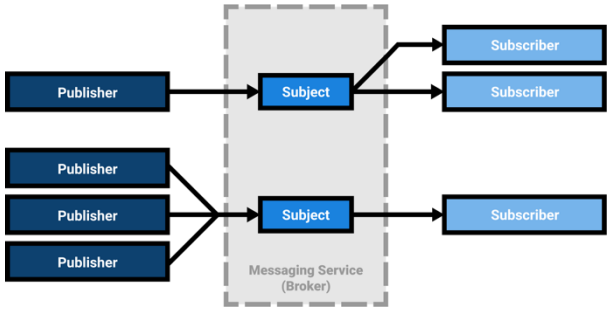
3.3 UDP (User Datagram Protocol)

UDP es un protocolo no orientado a la conexión. No establece un canal persistente; cada mensaje es un datagrama independiente. Sus principales características son:

- No confiable: los mensajes pueden perderse o llegar duplicados.
- No garantiza orden: los paquetes pueden llegar desordenados.
- Mayor velocidad y menor overhead, ideal para streaming o videojuegos

3.4 Modelo Publicación - Suscripción

El modelo de publicación y suscripción es un patrón de comunicación asíncrona basado en mensajes, en el cual estos se intercambian de forma anónima a través de un intermediario (broker). Las aplicaciones que producen información (publicadores) ponen esos datos a disposición en temas específicos que publican al broker. Por su parte, las aplicaciones que requieren información (suscriptores) se inscriben a esos temas para recibir los mensajes distribuidos por el intermediario, lo que permite una comunicación desacoplada y escalable entre los componentes de software. El broker actúa como un nodo central encargado de gestionar la distribución de los mensajes entre publicadores y suscriptores, aportando flexibilidad y robustez al sistema.



4. INFORMACIÓN BÁSICA

4.1 Reto propuesto

SISTEMA DE NOTICIAS DEPORTIVAS EN TIEMPO REAL.

Imagina que eres parte del equipo de desarrollo de una aplicación de streaming deportivo. El sistema debe permitir a los aficionados seguir en tiempo real los partidos de fútbol de su interés, recibiendo actualizaciones de los eventos más importantes (goles, cambios de jugadores, tarjetas, etc.).

Para lograrlo, se implementará un modelo de comunicación **publicación–suscripción** en el que:

- Los **publicadores (Publishers)** representan a periodistas deportivos que reportan en vivo lo que ocurre en el partido.
- El **broker** funciona como un canal de distribución central que recibe todos los mensajes de los publicadores y los reenvía a los aficionados interesados.
- Los **suscriptores (Subscribers)** son los hinchas que siguen uno o varios partidos y reciben las actualizaciones en tiempo real.

4.2 Descripción del Sistema

El modelo está compuesto por tres elementos principales:

- **Publisher (Publicador):**
  - Genera mensajes asociados a un tema específico (ej. un partido en particular).
  - Ejemplos de mensajes:
    - "Gol de Equipo A al minuto 32"
    - "Cambio: jugador 10 entra por jugador 20"
    - "Tarjeta amarilla al número 10 de Equipo B"
- **Broker:**
  - Recibe todos los mensajes provenientes de los publicadores.
  - Redistribuye cada mensaje a los suscriptores interesados en ese partido.
  - No modifica el contenido de los mensajes, únicamente se encarga de enviarlos al público correcto.
- **Subscriber (Suscriptor):**
  - Se suscribe a uno o varios partidos de su interés.
  - Recibe en su pantalla las actualizaciones en vivo enviadas por el broker.
  - Ejemplo: si un suscriptor sigue el partido *Equipo A vs Equipo B*, al ocurrir un gol recibirá el mensaje "Gol de Equipo A al minuto 45".

El laboratorio constará de **dos versiones del sistema**, que permitirán comparar el comportamiento de diferentes protocolos de transporte: una versión donde la comunicación entre broker, publicadores y suscriptores se implementa usando sockets TCP, y otra donde esa misma comunicación se implementa usando sockets UDP.

5. PROCEDIMIENTO

5.1 Configuración inicial

1. Todos los programas deben implementarse en lenguaje C. Se recomienda compilar con `gcc` y ejecutar las pruebas en un entorno **Linux**.
   Nota: no se aceptarán implementaciones en otros lenguajes, ya que el objetivo principal es practicar la programación de sockets en C.

2. Cree un directorio del laboratorio

3. Dentro del directorio, organice los archivos en:

   - `broker_tcp.c, publisher_tcp.c, subscriber_tcp.c`
   - `broker_udp.c, publisher_udp.c, subscriber_udp.c`

5.2 Pruebas y Comparación

1. Ejecute simultáneamente al menos **un** broker, **dos** suscriptores y **dos** publicadores.

2. Use Wireshark para capturar tráfico en los puertos definidos.

3. Analice:

   - En TCP: ¿los mensajes llegan completos y en orden? ¿Cómo maneja TCP la confiabilidad y el control de flujo?
   - En UDP: ¿qué evidencias hay de pérdida o desorden en los mensajes?
   - ¿Qué diferencias observa en el manejo de la conexión entre ambos protocolos?

   Nota: las respuestas deben ir acompañadas de evidencia gráfica (capturas de pantalla, gráficas de Wireshark, etc.).

6. TRABAJO PROPUESTO

1. Implementar el sistema **publicador–suscriptor** en TCP y en UDP.

2. **No** está permitido el uso de librerías de implementación de sockets a menos que se especifique punto a punto la interacción del programa con cada función de la librería, debe estar totalmente documentado.

3. Enviar al menos **10 mensajes** desde cada publicador y verificar que llegan correctamente a los suscriptores suscritos.

4. Capturar el tráfico con **Wireshark** y guardar al menos dos archivos:

- tcp_pubsub.pcap
- udp_pubsub.pcap

5. Comparar el desempeño de TCP y UDP en una tabla considerando los siguientes criterios:

   - Confiabilidad
   - Orden de entrega
   - Pérdida de mensajes
   - Overhead de cabeceras/protocolo

6. Responder a las siguientes preguntas de análisis:

   - ¿Qué ocurriría si en lugar de dos publicadores (partidos transmitidos) hubiera cien partidos simultáneos? ¿Cómo impactaría esto en el desempeño del broker bajo TCP y bajo UDP?
   - Si un gol se envía como mensaje desde el publicador y un suscriptor no lo recibe en UDP, ¿qué implicaciones tendría para la aplicación real? ¿Por qué TCP maneja mejor este escenario?
   - En un escenario de seguimiento en vivo de partidos, ¿qué protocolo (TCP o UDP) resultaría más adecuado? Justifique con base en los resultados de la práctica.
   - Compare el overhead observado en las capturas Wireshark entre TCP y UDP. ¿Cuál protocolo introduce más cabeceras por mensaje? ¿Cómo influye esto en la eficiencia?
   - Si el marcador de un partido llega desordenado en UDP (por ejemplo, primero se recibe el 2–1 y luego el 1–1), ¿qué efectos tendría en la experiencia del usuario? ¿Cómo podría solucionarse este problema a nivel de aplicación?
   - ¿Cómo cambia el desempeño del sistema cuando aumenta el número de suscriptores interesados en un mismo partido? ¿Qué diferencias se observaron entre TCP y UDP en este aspecto?
   - ¿Qué sucede si el broker se detiene inesperadamente? ¿Qué diferencias hay entre TCP y UDP en la capacidad de recuperación de la sesión?
   - ¿Cómo garantizar que todos los suscriptores reciban en el mismo instante las actualizaciones críticas (por ejemplo, un gol)? ¿Qué protocolo facilita mejor esta sincronización y por qué?
   - Analice el uso de CPU y memoria en el broker cuando maneja múltiples conexiones TCP frente al manejo de datagramas UDP. ¿Qué diferencias encontró?
   - Si tuviera que diseñar un sistema real de transmisión de actualizaciones de partidos de fútbol para millones de usuarios, ¿elegiría TCP, UDP o una combinación de ambos? Justifique con base en lo observado en el laboratorio.

7. ENTREGABLES

El entregable para este laboratorio incluye:

1. Un informe en formato PDF que contenga:

   - Evidencia clara de los resultados (capturas de pantalla, fragmentos de código, gráficos de análisis, etc.) con títulos y descripciones.

- El informe debe incluir las respuestas a las preguntas planteadas.
- Un enlace de descarga (Google Drive, GitHub, etc.) para acceder a los archivos de captura (.pcap).

2. Una carpeta comprimida con todos los archivos fuente en C (.c) debidamente organizados y comentados.

HISTORIAL DE CAMBIOS

Registro de cambios

Fecha	Autor(es)	Versión	Referencia de cambios
05/09/2025	**Nathalia Quiroga** n.quiroga@uniandes.edu.co	1	Creación