I310 - Redes y Comunicación

Trabajo Práctico D: Desarrollo de aplicaciones distribuidas con BSD Sockets

Ingeniería en Inteligencia Artificial
Universidad de San Andrés -
Noviembre 2025

Ignacio Gremes
Timoteo Menceyra
Felipe Viaggio

Protocolo de Transferencia de Archivos UDP

- Cliente → Servidor (upload de archivos)
- Stop & Wait: espera ACK antes de enviar siguiente paquete

4 Fases Obligatorias:

1. HELLO - Autenticación con credenciales 2. WRQ - Solicitud de escritura (Write Request)
3. DATA - Transferencia con alternancia seq_num (0,1,0,1...)
4. FIN - Finalización de sesión

Estructura del Proyecto

protocol.h

- PDU (Formato de paquetes)
- ClientState (Estado del Cliente)
- ClientSession (Sesiones en Servidor)

client.c

- init_client()
- send_hello()
- send_wrq()
- send_file_data()
- send_fin()

server.c

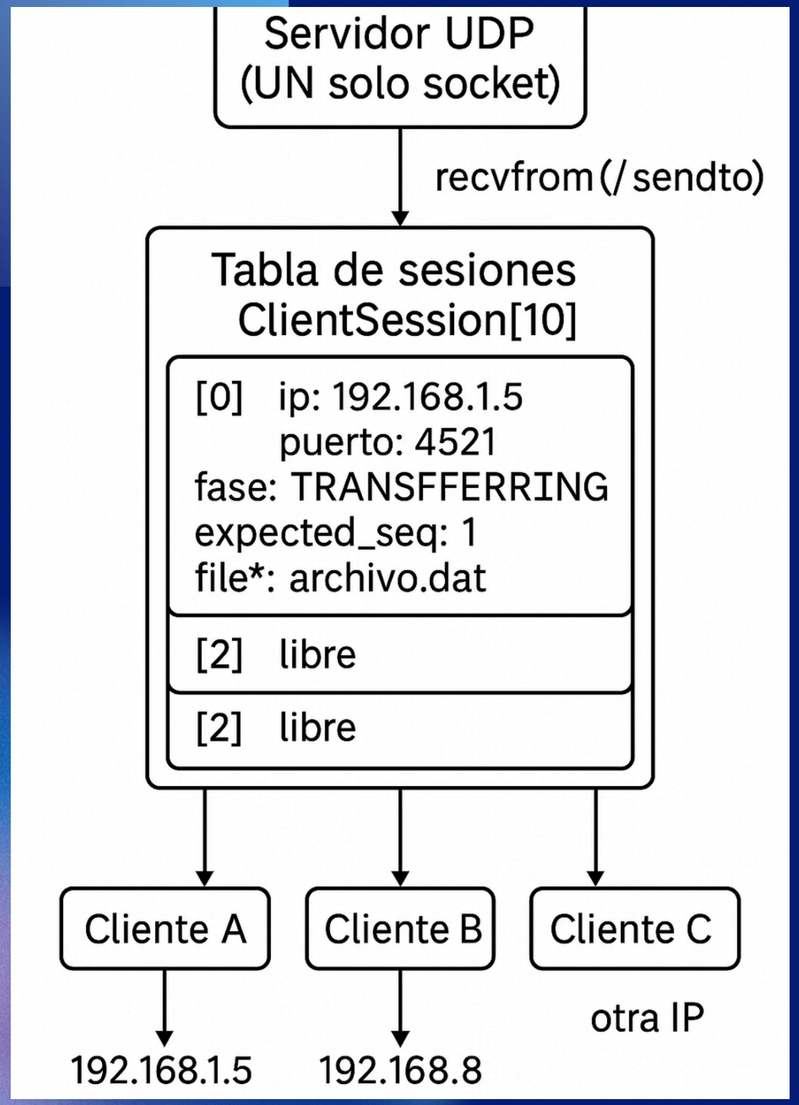
- handle_hello()
- handle_wrq()
- handle_file_data()
- handle_fin()
- concuerrencia

¿Como manejar la concuerrencia?

Cada sesión mantiene:
- Dirección del cliente (IP:Puerto)
- Fase actual del protocolo
- Seq_num esperado (0 o 1)
- FILE* del archivo siendo escrito

¿Por qué no fork() como en TCP?

- En UDP todos los clientes comparten el mismo socket.



UDP

Desafios

A lo largo del desarrollo del proyecto, nos topamos con muchos desafios, entre ellos algunos de estos:

- Stop & Wait con Timeouts
- Validaciones de Entrada
- Retransmisiones
- Descarte Silencioso

Resultado Y Aprendizajes

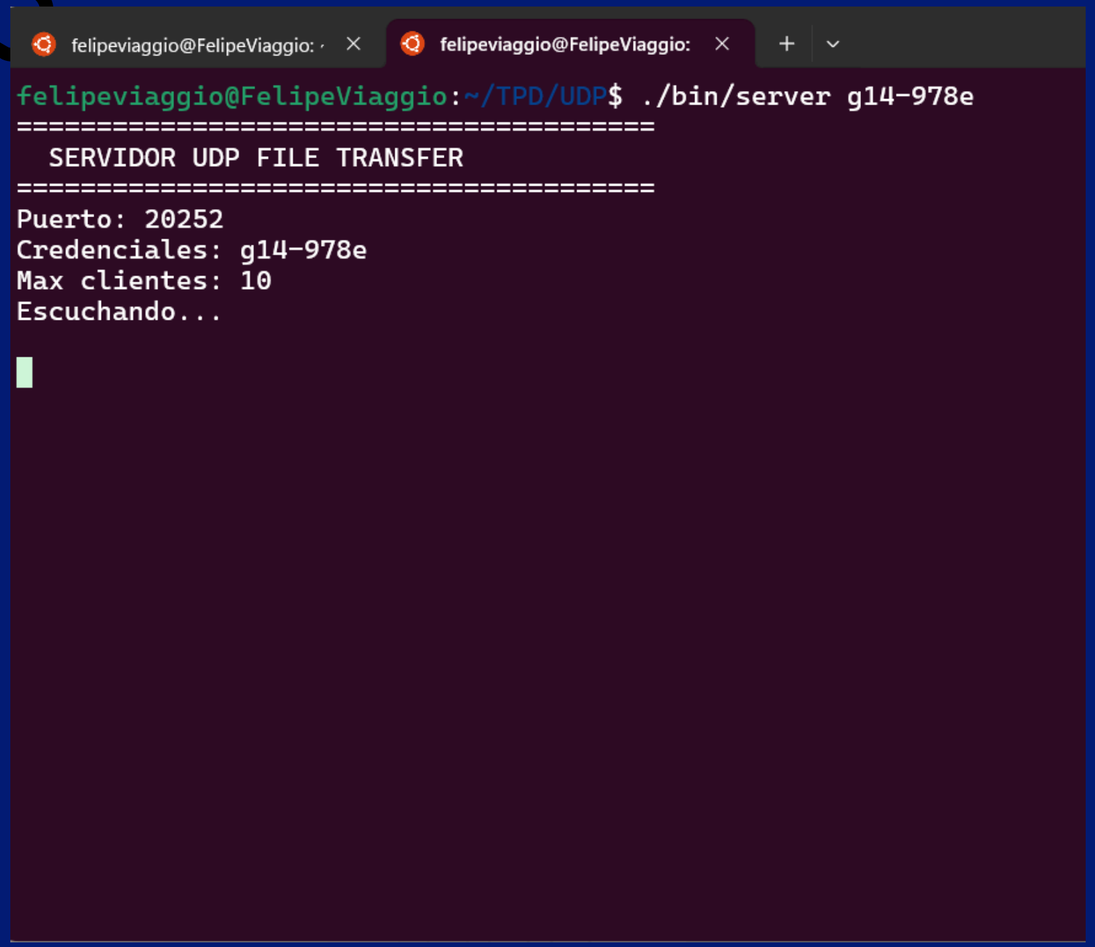
✓ Validación exitosa con archivo g14.data
MD5: 725d0fe110bcbd1f8f225cb04fd6e54d

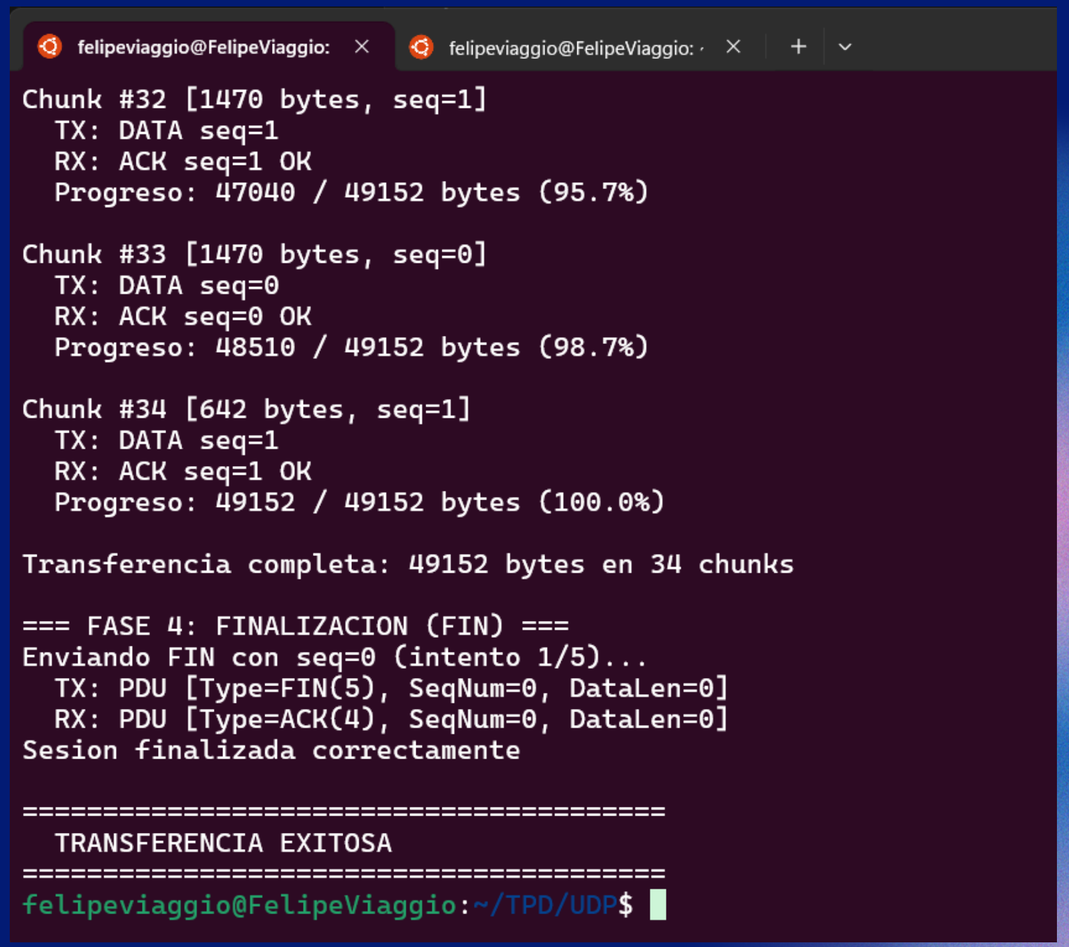
Performance con archivo de 20Kb

Escenario	Tiempo
LAN (2ms RTT)	✅ 168ms
Internacion (180ms RTT)	⚠️ 3,34 s
Lunar (2.56s RTT)	❌ 44 s

Conclusión: Stop & wait es simple pero ineficiente con latencia alta.

Protocolo en Acción





TCP Performance

Objetivo: medir one-way delay entre cliente y servidor mediante BSD Sockets

Estructura de las PDU

Timestamp

- Momento en el que sucedió el evento

Payload

- Carga útil del mensaje

Delimiter

- Caracter especial que marca el fin del mensaje

Cliente

Activación del cliente

./cliente_tcp -d __ N __

Funcionamiento

Genera PDU de tamaño aleatorio

Escribe el timestamp de origen

Agrega el delimitador ‘|’ en el último byte

Servidor

Funcionamiento

Recibe PDU, reensambla stream y valida tamaño

Con la PDU completa, calcula delay = arrival_ts - origin_ts

Guarda delays en un archivo CSV

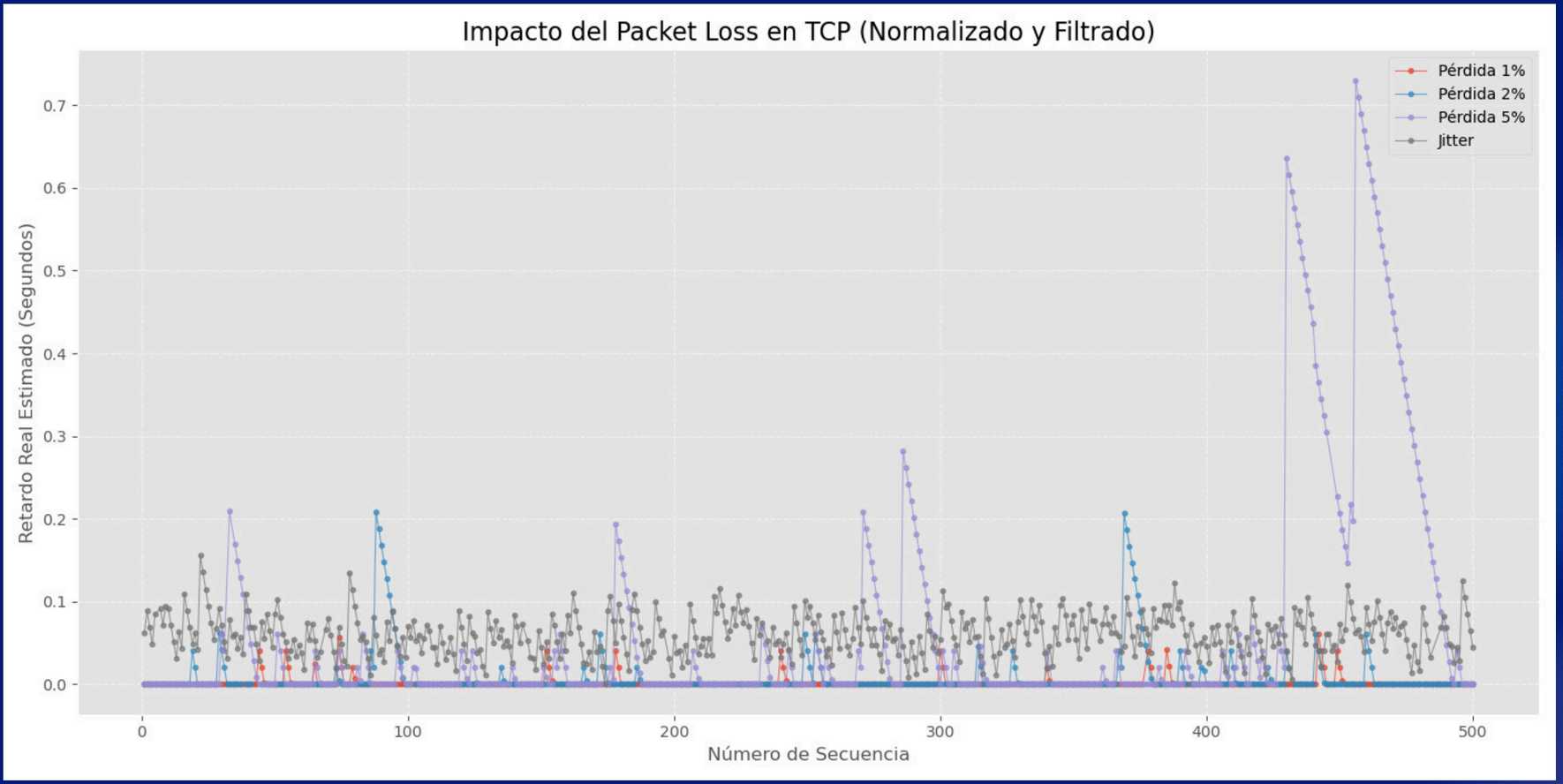
Robustez ante lecturas parciales

Un buffer de ensamblado soluciona este problema

CSV

1	seq,delay_seconds
2	1,0.00131
3	2,0.00020
4	3,0.00018
5	4,0.00017
6	5,0.00018
7	6,0.00019
8	7,0.00019
9	8,0.00015
10	9,0.00019
11	10,0.00016
12	11,0.00022
13	12,0.00016
14	13,0.00023
15	14,0.00023
16	15,0.00023
17	16,0.00018
18	17,0.00024
19	18,0.00024
20	19,0.00021
21	20,0.00017
22	21,0.00015
23	22,0.00015
24	23,0.00011
25	24,0.00017
26	25,0.00019
27	26,0.00015

Packet Loss


Impacto del Packet Loss en TCP (Normalizado y Filtrado)
Pérdida 1%
Pérdida 2%
Pérdida 5%
Jitter
Retardo Real Estimado (Segundos)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
100
200
300
400
500
Número de Secuencia

Conclusiones