Licenciatura en Ciencias de la computación

Sistemas Operativos 1

# Sistema P2P Simple para Compartir Archivos en LAN

Integrantes del grupo

Jano Vaquer

Mario Mérida

Entrega

29/07/2025

## 1. Introducción

El sistema se compone de nodos P2P que pueden descubrirse entre sí, anunciar archivos disponibles, buscar archivos de otros nodos y descargarlos. Cada nodo es un proceso principal que se encarga de ejecutar los servicios del sistema de forma concurrente. Es decir, el programa se inicializa y crea procesos que comparten tiempo de vida con el proceso principal. Estos procesos se encargan de responder solicitudes de clientes conectados y mantener actualizado el registro del nodo dentro del sistema.

## 2. Descripción General del Sistema

El sistema se compone de nodos P2P que pueden descubrirse entre sí, anunciar archivos disponibles, buscar y descargar archivos. Cada nodo se implementa como un proceso principal que lanza otros procesos concurrentes para ejecutar los distintos servicios del sistema. Es decir, el programa sigue los lineamientos del trabajo práctico y crea procesos secundarios que comparten tiempo de vida con el proceso principal. Estos se encargan de responder solicitudes de clientes conectados y de mantener actualizado el registro del nodo dentro del sistema.

## 3. Módulos principales:

Modularizamos el funcionamiento del programa de la siguiente manera:

- config.hrl: archivo de parámetros de configuración del sistema. Define, por ejemplo, los caracteres aceptados para identificar un nodo y los puertos de conexión utilizados.
- nodo.erl: programa principal que inicializa los módulos y crea procesos concurrentes para el funcionamiento del sistema.
- archivos.erl: módulo encargado de la gestión de archivos y carpetas locales.
- name.erl: módulo que genera y confirma un nombre único para identificar al nodo dentro del sistema.
- servidor.erl: módulo encargado de recibir mensajes como DOWNLOAD_REQUEST y SEARCH_REQUEST de otros nodos conectados.
- udp.erl: módulo que envía mensajes HELLO para anunciar la presencia del nodo y también recibe mensajes HELLO de otros nodos para registrar sus identificadores.
- cli.erl: módulo de interacción con el usuario. Implementa la interfaz de línea de comandos (CLI) del nodo.

## 4. Como ejecutar el programa

Para ejecutar el programa primero compilar con "make". Luego abrir erlang con "erl" y ejecutar mediante "nodo:inicio()."

### 4.1 Comandos para Makefile

- make → compila todos los ".erl"
- make clean → borra los ".beam"
- make dumpclean → borra los ".dump"

### 4.2 Comandos para de interacción con sistema "ClienteP2P>":

- nodoID →Nombre del nodo
- lista → Lista archivos locales compartidos
- buscar <patrón> → Busca archivos en la red
- descargar <archivo> <nodo>→Descarga un archivo de otro nodo
- ayuda→Muestra esta ayuda
- exit→Finaliza el nodo

## 5. Diseño e Implementación

### 5.1 Concurrencia

Se eligió Erlang por su capacidad para generar procesos concurrentes y livianos, ya que emplea un modelo de actores basado en paso de mensajes. Además, está diseñado específicamente para sistemas robustos y tolerantes a fallos.

### 5.2 Protocolo de Comunicación

Se utiliza UDP para los mensajes iniciales de descubrimiento, como HELLO y NAME_REQUEST. Para las operaciones principales, como búsqueda, respuesta, solicitudes de descarga y transferencia de archivos (divididos en chunks cuando superan los 4MB), se emplea TCP.

## 6. Robustez

Se logró una implementación robusta gracias al uso de Erlang, que facilita la gestión de concurrencia y errores, sumado al manejo eficiente de sockets de comunicación TCP y UDP mediante funciones como open, send, listen, entre otras. Además, se implementó la detección y eliminación de nodos inactivos, utilizando la gestión de timestamps almacenados en ETS para asegurar la actualización y consistencia del sistema.

Con respecto al registro de nodos según el consenso definido en por el curso utilizamos ETS. Este es un módulo de Erlang que garantiza el acceso, eliminación, o búsqueda de tuplas como operaciones atómicas. Asimismo, este módulo no genera errores si intentáramos conectarnos con un nodo aún en los registros.

Como Erlang maneja la salida por pantalla con un group leader en el caso de SEARCH_REQUEST no hace falta recopilar los datos en un proceso, sino que el io:format lo maneja adecuadamente.

## 7. Conclusión

La concurrencia se maneja de forma sencilla mediante las primitivas spawn y receive. Erlang resulta ideal para sistemas distribuidos y tolerantes a fallos, gracias a su modelo de procesos ligeros y su robusto manejo de errores. El sistema cumple con los requerimientos funcionales y de robustez planteados en el enunciado del trabajo práctico.