DistriShare

P2P HÍBRIDA

Descripción del proyecto y objetivos.

- **DistriShare** es una aplicación de compartición de archivos diseñada como una **arquitectura P2P híbrida**

- Combinando
  - un servidor centralizado (Bootstrap Server)
  - un mecanismo distribuido (descubrimiento UDP por multicast).

Arquitectura final implementada (1)

Tres capas principales:

- Servidor Bootstrap (Bootstrap Server): registro de nodos

- Descubrimiento híbrido de nodos
  - **Descubrimiento vía Bootstrap**:
  - **Descubrimiento vía Multicast UDP (Manual)**:

- Transmisión y gestión de archivos

Arquitectura final implementada (2)

- Descubrimiento híbrido de nodos
  - **Descubrimiento vía Bootstrap**:
    - Al arrancar, cada peer puede invocar register_with_bootstrap(ip, port)
    - registrarse en el bootstrap y recibir la lista de nodos conocidos. known_nodes
  - **Descubrimiento vía Multicast UDP (Manual)**:
    - Cada peer, al invocar peer.start_multicast(), crea una instancia de MulticastDiscovery
    - Inician 2 hilos: Sender (HELLO ip puetro a 224.1.1.1:10000) y Listener(Recibe HELLO ip puerto)
    - Detener Multicast envia GOODBYE ip puerto

Arquitectura final implementada(3)

- **Transmisión y gestión de archivos**
  - **Buscar un archivo**: busca en known_nodes
    - se invoca remote_file_exists(ip, port, filename) (envía JSON {"type":"search","filename":..., "secret":...})
    - Si tiene el archivo en shared_files/, responde "FOUND" y el buscador registra esa dirección.
  - **Descargar un archivo**: tras conocer (ip, port) donde existe,
    - el peer cliente envía {"type":"download","filename":..., "secret":...};
    - el servidor abre el fichero en shared_files/ y lo envía en trozos de 4096 bytes; el cliente lo escribe en downloads/<filename>.

Conceptos de computación distribuida aplicados

- Arquitectura P2P Híbrida
  - Combina un elemento centralizado (servidor Bootstrap) y un elemento totalmente distribuido (descubrimiento multicast).
- Descubrimiento de nodos y tolerancia a fallos
  - **Descubrimiento por multicast UDP**
  - Detección de nodos vivos
  - Tolerancia a fallos
- Indexación y búsqueda distribuida
- Transferencia de archivos en paralelo
  - El servidor TCP local (start_file_server) atiende conexiones en un hilo independiente: por cada cliente entrante, lanza un nuevo hilo
  - Los archivos se envían en bloques de 4096 bytes,

Pasos y procedimientos

- encender servidor Bootstrap
  - python3 -m network.bootstrap_server
- ejecutar mediante cli
  - python3 -m cli.main 9000
  - python3 -m cli.main 9001
  - python3 -m cli.main 192.168.1.42 9100
- ejecutar web
  - python -m web.app --peer-port 9000 --flask-port 5000
  - python -m web.app --peer-port 9002 --flask-port 5002
  - python -m web.app --peer-ip 90.167.87.98 --peer-port 9002 --flask-port 5002

Retos encontrados y soluciones implementadas

- Repaso de conceptos de redes en protocolos
- Problema con el caso de uso de Internet, solución: Port Forwarding
- Fallas parciales durante transferencia de archivos
  - Cuando llega el fragmento si sale vacío, fin trasnferencia.
  - Si recibe error, borra el archivo parcial
  - Implementación del timeout, si expira, cierra conexión y borra.
- Problema con lista de nodos
  - Tres listas de nodos: todo, Bootstrap, Multicast
- Difícil manejo en comando
  - Versión web sencillo con Flask Web

Conclusiones y posibles líneas de trabajo futuras

- Me obligó aprender conceptos para hacer la practica por mi cuenta y eso me motiva el aprendizaje, ya que no es lo mismo aprender obligados que aprender por propia cuanta para solucionar un problema.
- Posibles mejoras:
  - Pagina web UI más intuitivo
  - Seguridad
  - Persistencia de nodos en el Bootstrap(de momento solo en memoria)
  - Mejora de sistema de búsqueda
  - NAT (de momento solo Multicast local)