

Distributed Systems

Complete Name: Carlos Castro

Midterm

1 Arquitectura

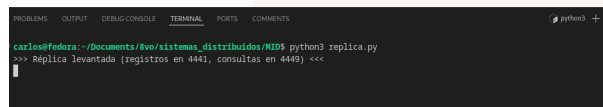
Hay un **main server** (server) que registra los servicios y una **replica** que almacena la copia del registro. El **publisher** registra su servicio en el server y este replica la información hacia la replica. El **subscriber** solicita la ubicación por *nombre del servicio*; si el server no responde, consulta a la replica. Los códigos usados estarán en el repositorio de Git: https://github.com/carloscastro4172/SISTEMAS_DISTIBUIDOS/tree/main/MID.

2 Evidencia

Añadiremos los puertos usados para facilitar el entendimiento del código además de las capturas correspondientes del funcionamiento:

- **Server:** registro en puerto 4431, lookup en puerto 4421, PUSH hacia la replica en puerto 4441.
- **Replica:** PULL en puerto 4441, lookup en puerto 4449.
- **Publisher:** registra en puerto 4431, publica en puerto 4432.
- **Subscriber:** consulta al server en puerto 4421 y, en caso de fallo, hace fallback a la replica en puerto 4449.

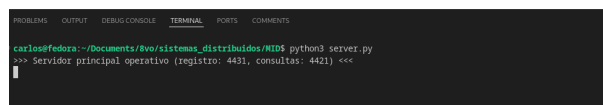
2.1 Replica encendida



```
carlos@fedora:~/Documents/8vo/sistemas_distribuidos/MID$ python3 replica.py
>>> Replica levantada (registros en 4441, consultas en 4449) <<<
```

Figure 1: Replica lista: recibe registros del server y atiende consultas de respaldo.

2.2 Main server encendido



```
carlos@fedora:~/Documents/8vo/sistemas_distribuidos/MID$ python3 server.py
>>> Servidor principal operativo (registro: 4431, consultas: 4421) <<<
```

Figure 2: Server listo para registro y resolución de nombres.

2.3 Publisher registra y publica

```
carlos@fedora:~/Documents/8vo/sistemas_distribuidos/MID$ python3 publisher.py
>>> Publisher solicitando registro en el servidor principal...
Publisher activo, enviando datos en tcp://localhost:4432
Mensaje emitido: TIME 2025-10-08 09:18:24
Mensaje emitido: TIME 2025-10-08 09:18:25
Mensaje emitido: TIME 2025-10-08 09:18:26
```

Figure 3: Registro del servicio en el server por parte del publisher.

```
carlos@fedora:~/Documents/8vo/sistemas_distribuidos/MID$ python3 replica.py
>>> Réplica levantada (registros en 4441, consultas en 4449) <<<
Réplica guardó el servicio: TIME -> tcp://localhost:4432
```

Figure 4: Replicación confirmada en la replica.

2.4 Subscriber obtiene dirección del server

```
pythcarlos@fedora:~/Documents/8vo/sistemas_distribuidos/MID$ python3 subscriber.py
>>> Escriba el nombre del servicio al que desea suscribirse: TIME
>>> Dirección recibida desde el principal: tcp://localhost:4432 <<<
>>> Suscripción establecida en tcp://localhost:4432. Esperando mensajes... <<<
>>> Mensaje recibido: TIME 2025-10-08 09:34:15 <<<
>>> Mensaje recibido: TIME 2025-10-08 09:34:17 <<<
```

Figure 5: Resolución por nombre con el server (dirección del publisher).

2.5 Caída del server (prueba de fallo)

```
carlos@fedora:~/Documents/8vo/sistemas_distribuidos/MID$ python3 subscriber.py
>>> Mensaje recibido: TIME 2025-10-08 09:40:12 <<<
carlos@fedora:~/Documents/8vo/sistemas_distribuidos/MID$ python3 subscriber.py
>>> Ingrese el nombre del servicio al que desea suscribirse: TIME
>>> Timeout. Falló el primario, intentando con replica... <<<
>>> Dirección obtenida del servidor replica: tcp://localhost:4432 <<<
>>> Suscripción establecida en tcp://localhost:4432, esperando mensajes... <<<
```

Figure 6: Failover: el subscriber obtiene la dirección desde la replica.

2.6 Mensajes del publisher tras el failover

```
carlos@fedora:~/Documents/8vo/sistemas_distribuidos/MID$ python3 subscriber.py
>>> Ingrese el nombre del servicio al que desea suscribirse: TIME
>>> Timeout. Falló el primario, intentando con replica... <<<
>>> Dirección obtenida del servidor replica: tcp://localhost:4432 <<<
>>> Suscripción establecida en tcp://localhost:4432, esperando mensajes... <<<
>>> Mensaje recibido: TIME 2025-10-08 09:40:20 <<<
>>> Mensaje recibido: TIME 2025-10-08 09:40:22 <<<
>>> Mensaje recibido: TIME 2025-10-08 09:40:24 <<<
>>> Mensaje recibido: TIME 2025-10-08 09:40:26 <<<
>>> Mensaje recibido: TIME 2025-10-08 09:40:28 <<<
```

Figure 7: Servicio continuo desde el publisher tras la caída del server.

3 Análisis breve

El **subscriber** resolvió el servicio **TIME** por nombre con el **server** y se conectó al **publisher**. Al detener el server, el subscriber reintentó contra la **replica** y obtuvo la misma dirección, manteniendo la recepción de mensajes. Esto evidencia tolerancia a fallos gracias a la replicación del registro de servicios. Limitación: si server y replica caen o si no hay sincronización de nuevos registros durante la caída, el servicio no se resuelve.

4 Orden de ejecución

1. `replica.py` → ver “Replica activa”.
2. `server.py` → ver “Server activo”.
3. `publisher.py` → registro y replicación del servicio.
4. `subscriber.py` (escribir `TIME`) → primero resuelve con el server; luego repetir tras matar el server para mostrar el *failover* hacia la replica.

