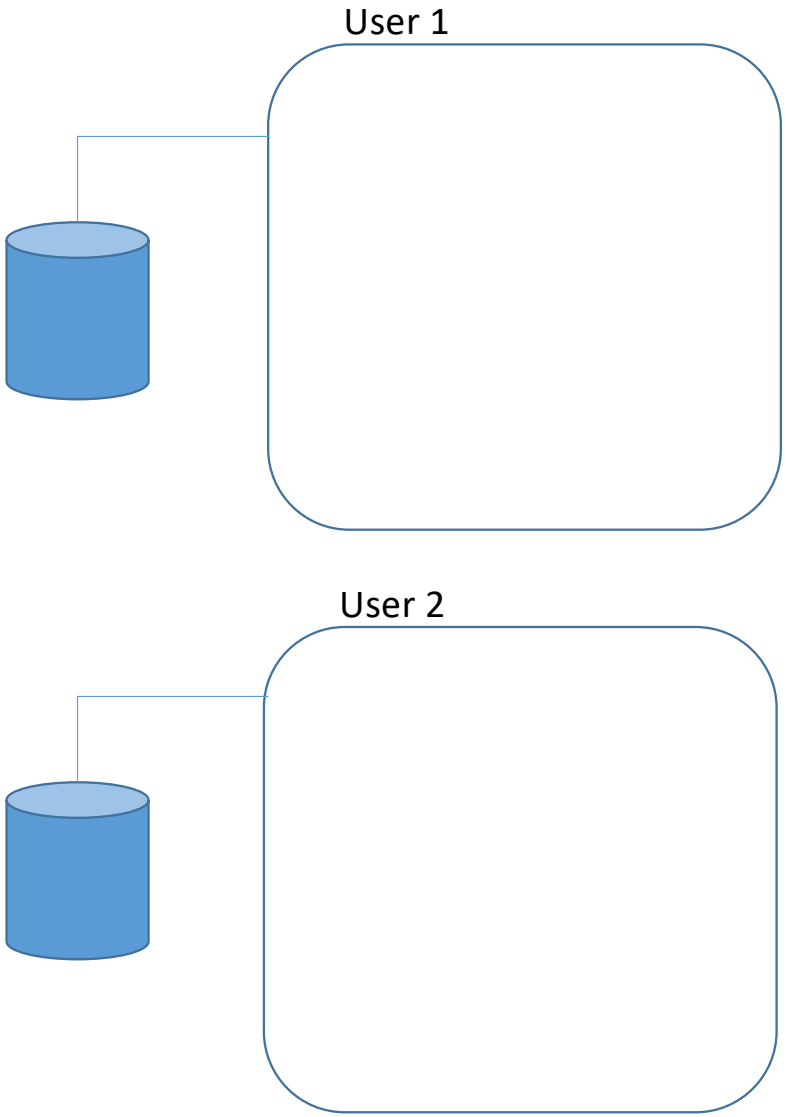
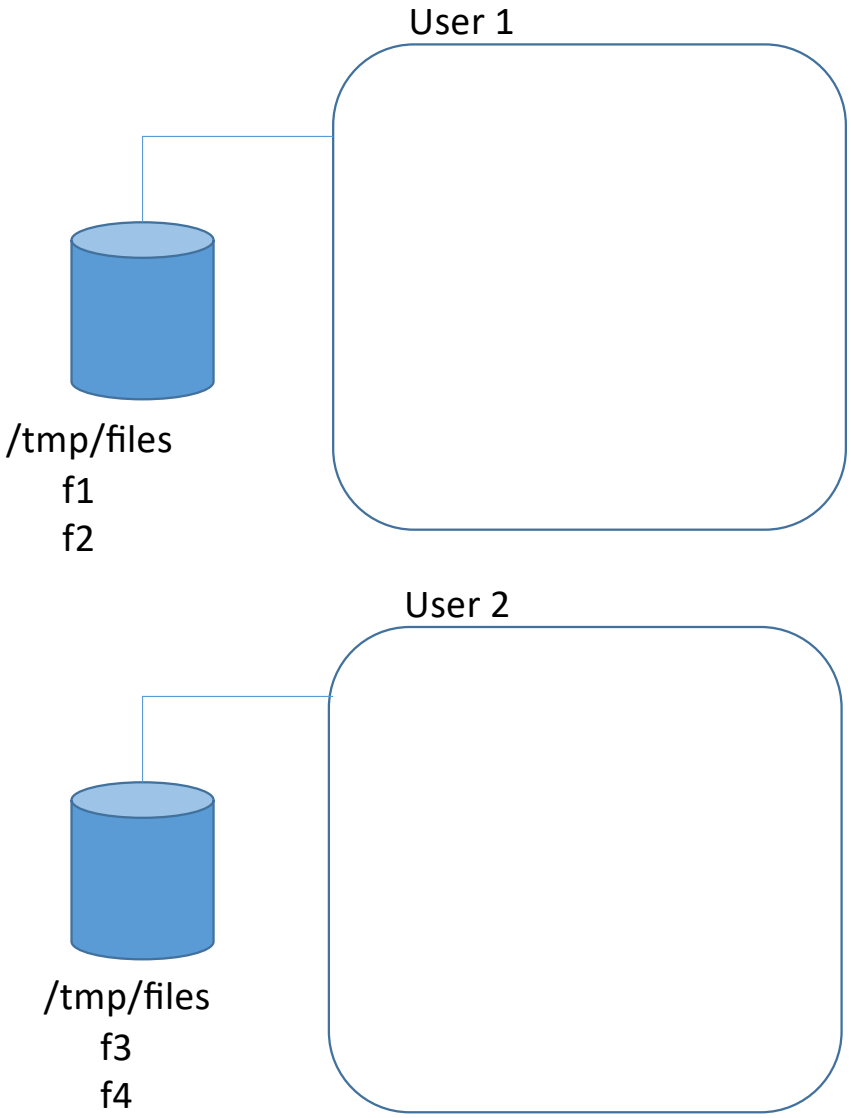
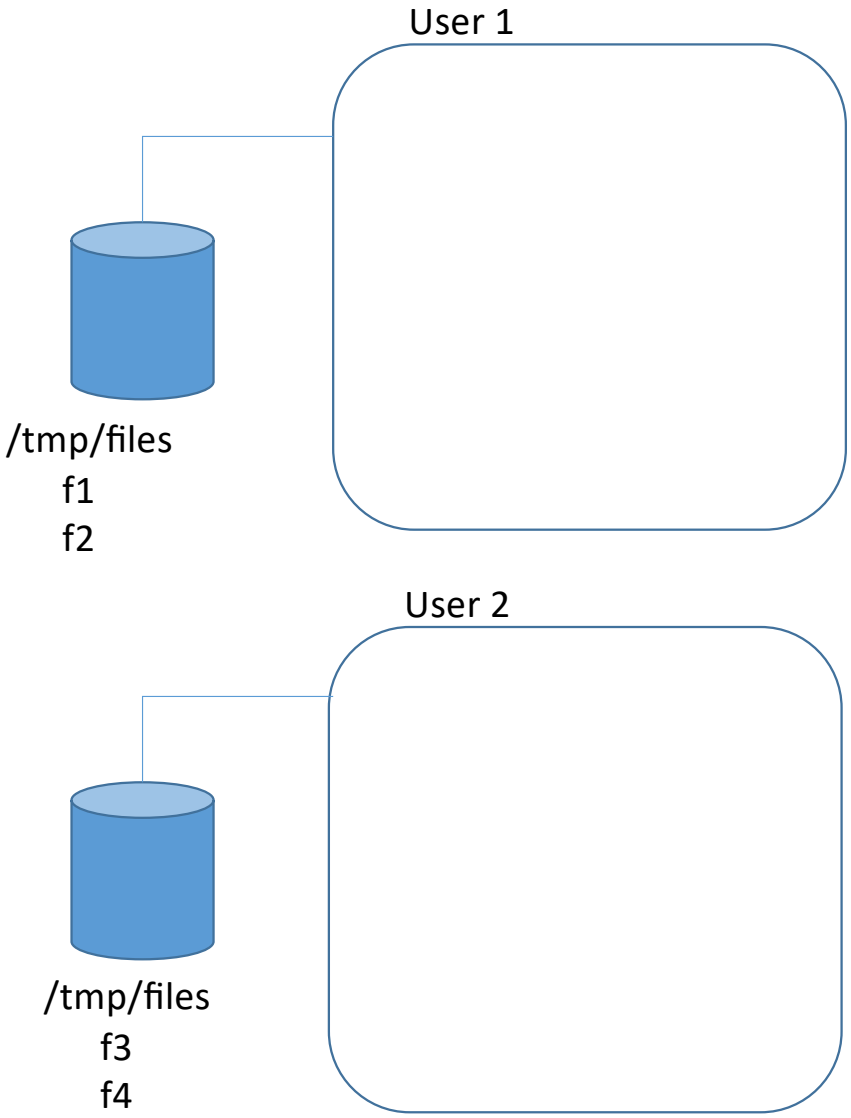
Sistemas Distribuidos

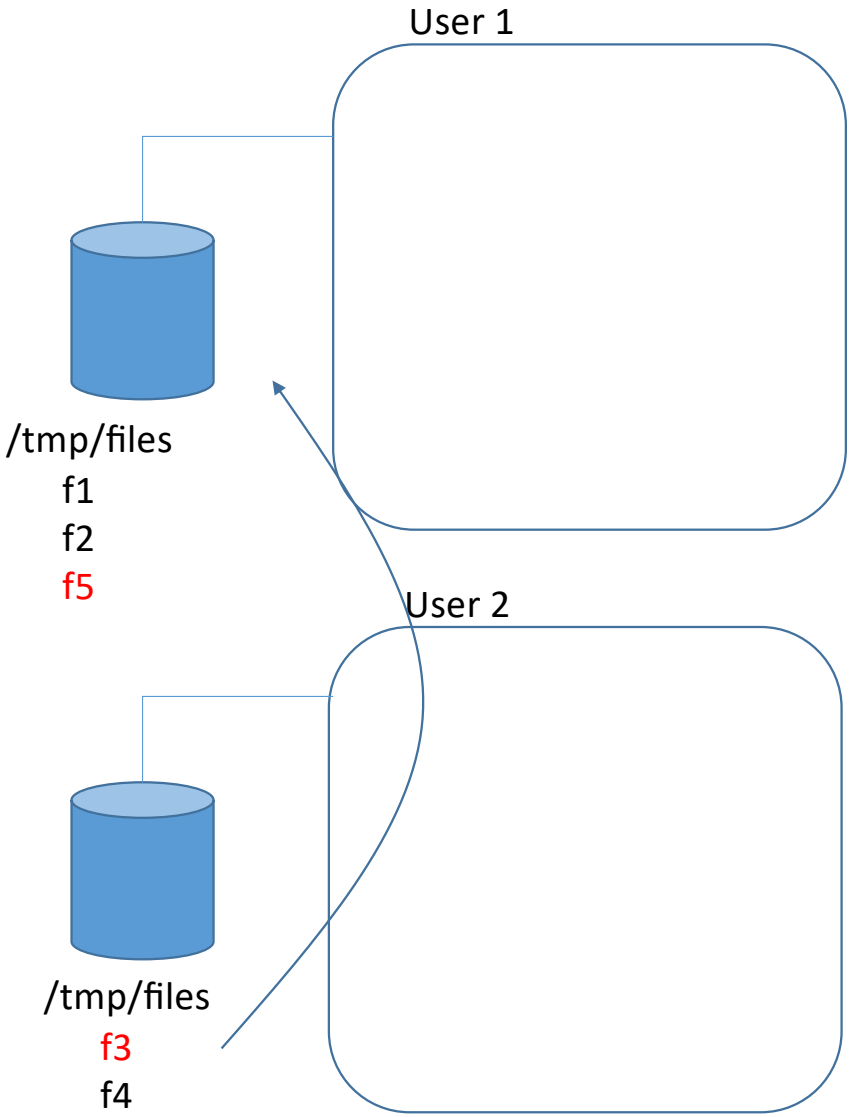
Proyecto:
Diseño e implementación de un sistema *peer-to-peer*


User 1
User 2

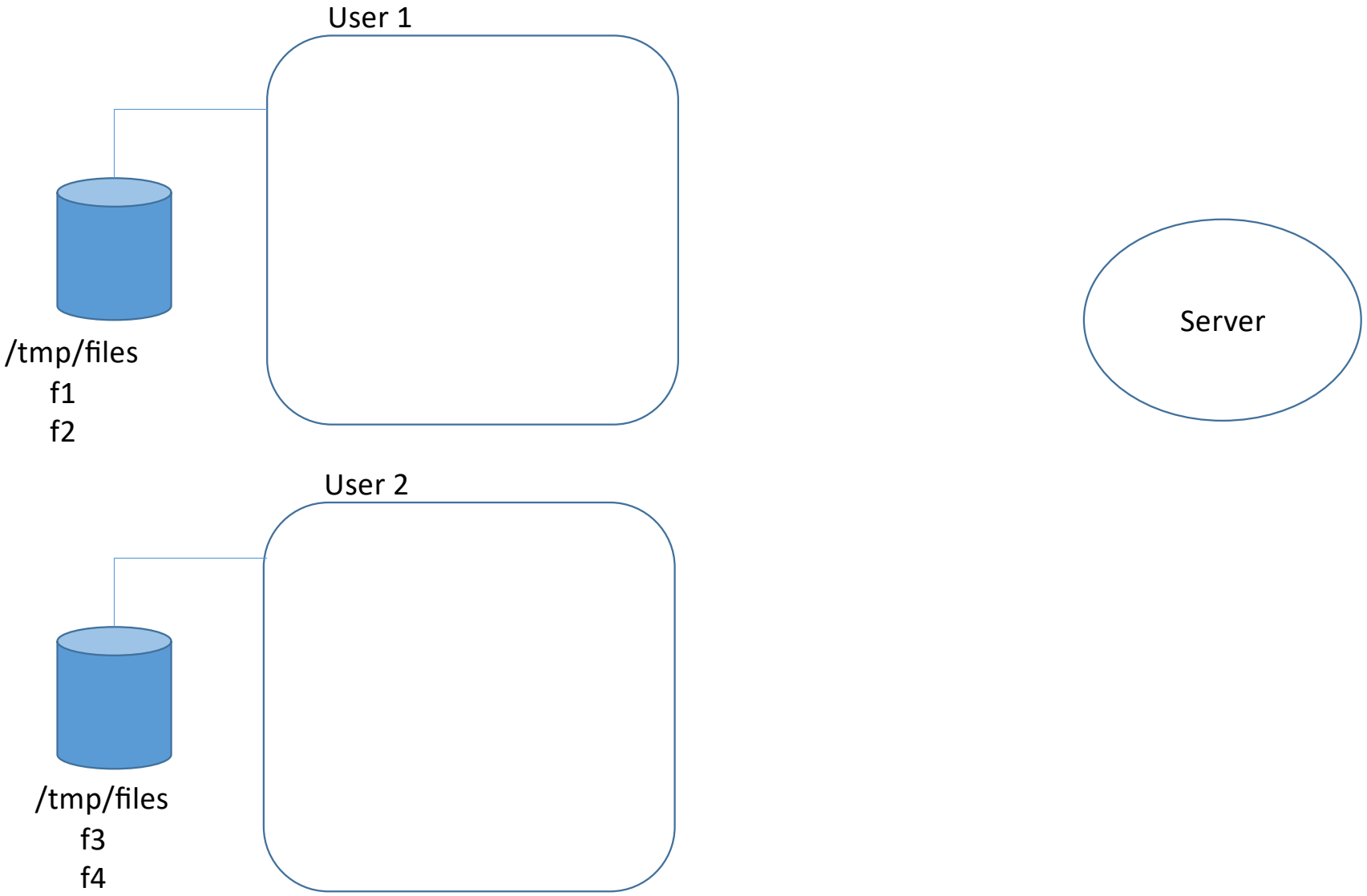

User 1
/tmp/files
f1
f2
User 2
/tmp/files
f3
f4

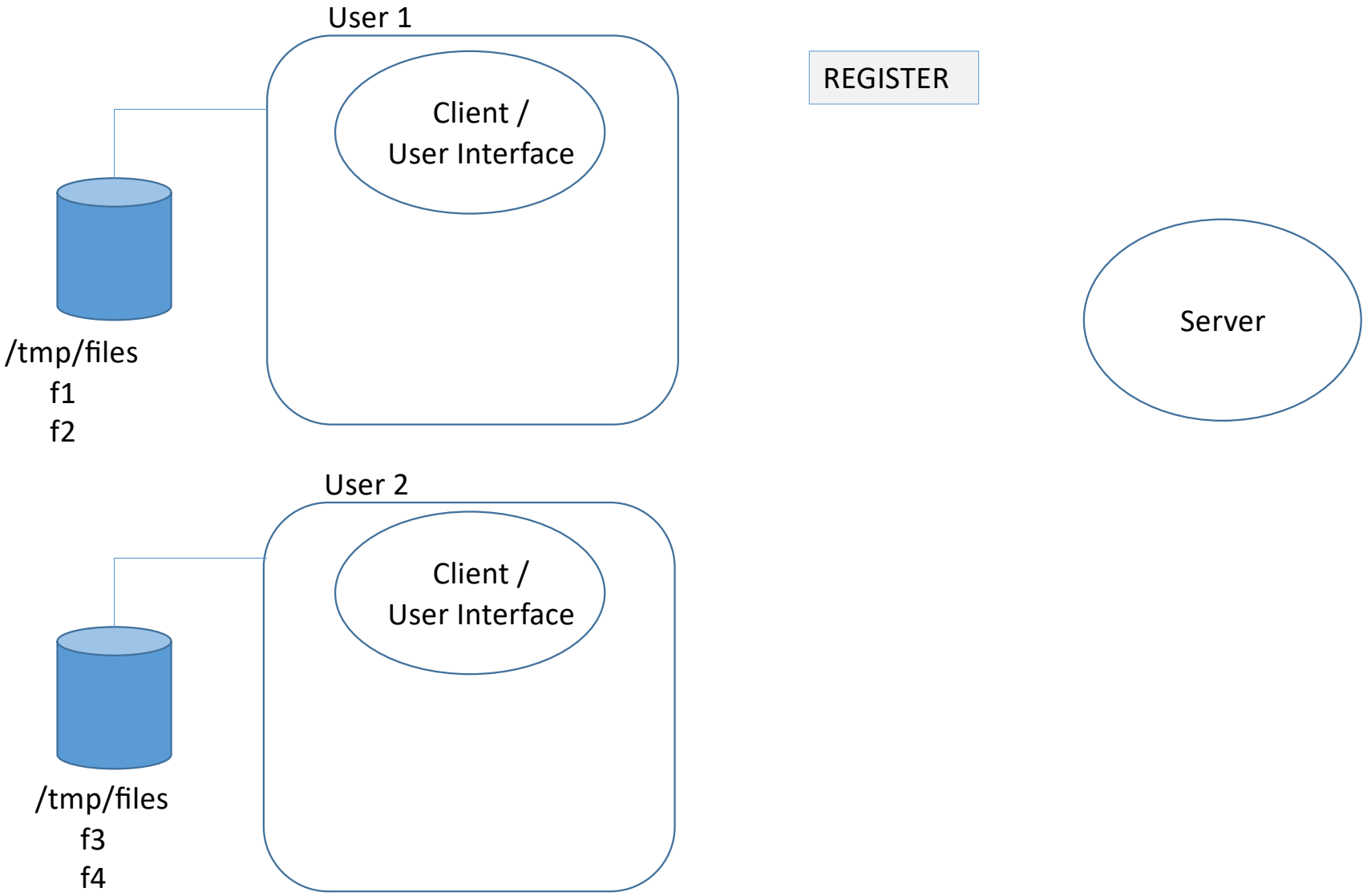


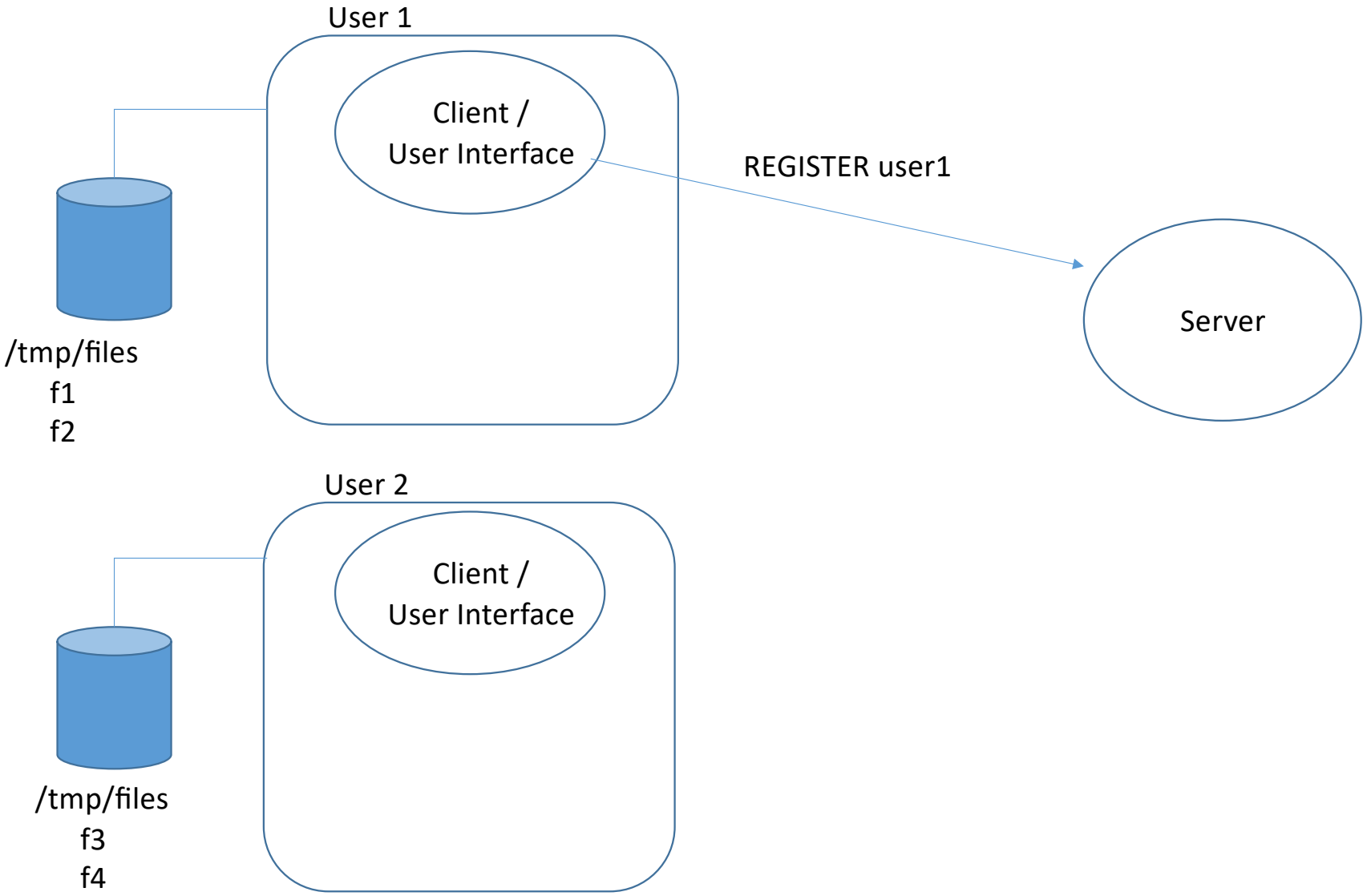
OBJETIVO: compartir y transferir ficheros entre usuarios/as

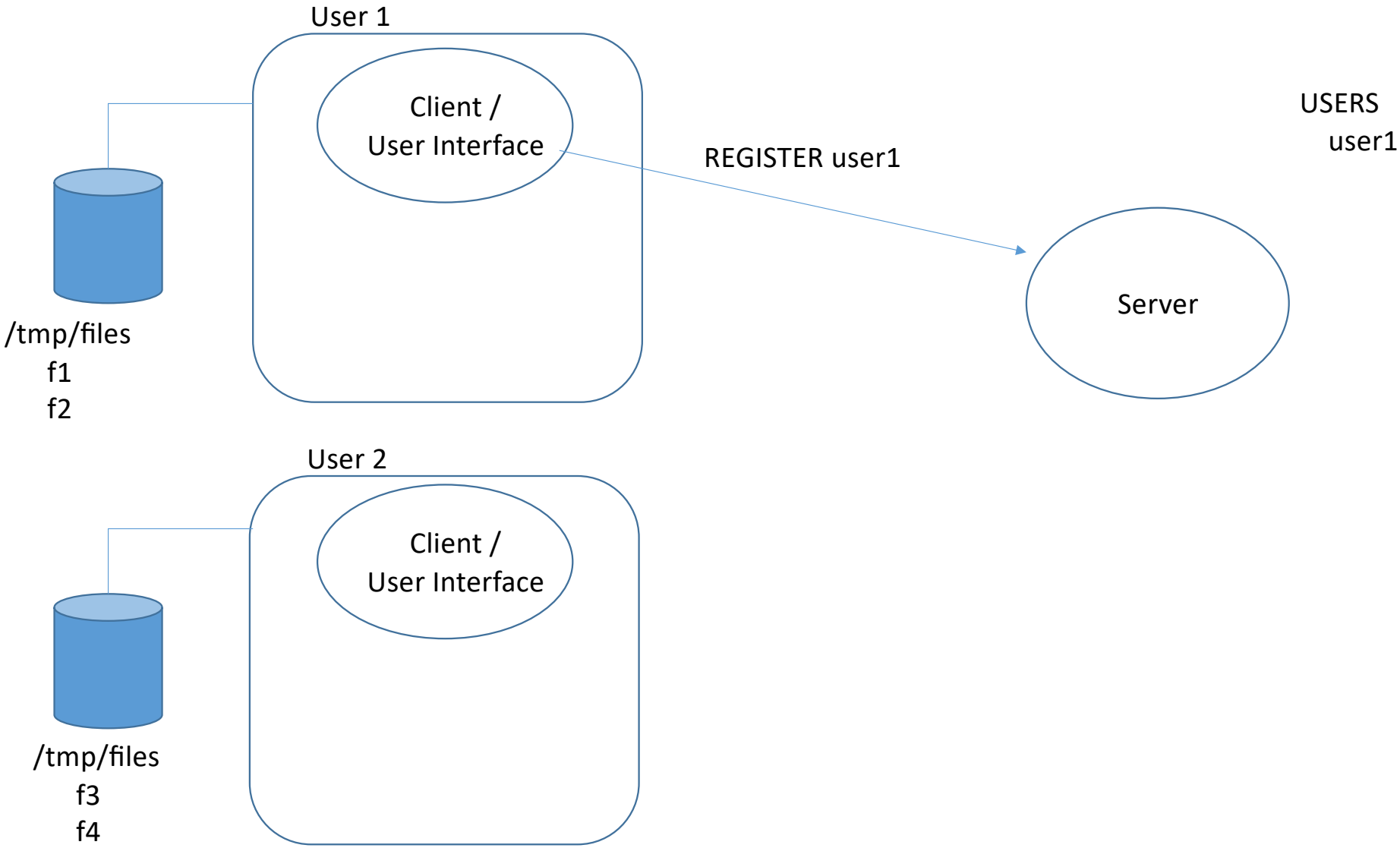


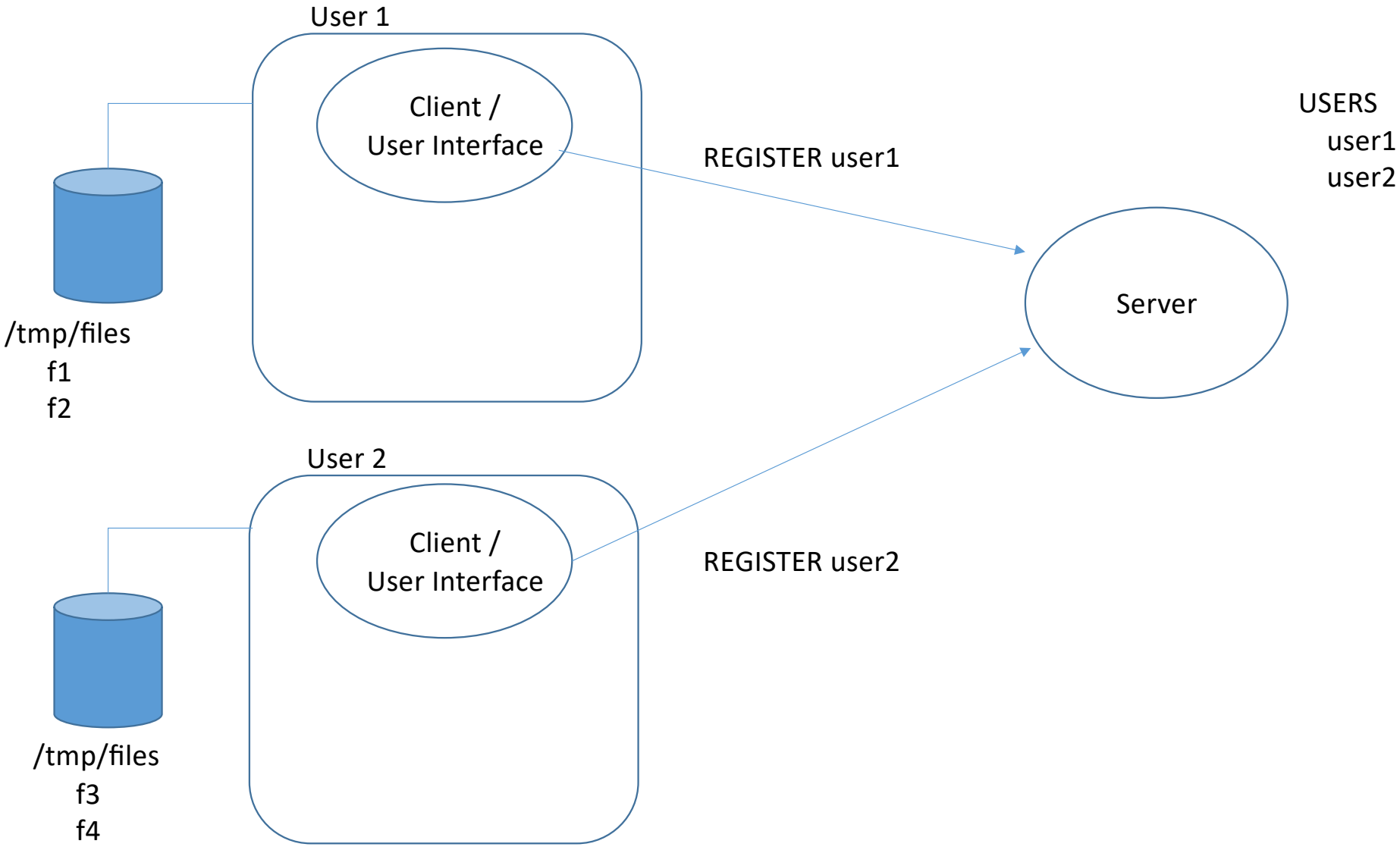
OBJETIVO: compartir y transferir ficheros entre usuarios/as

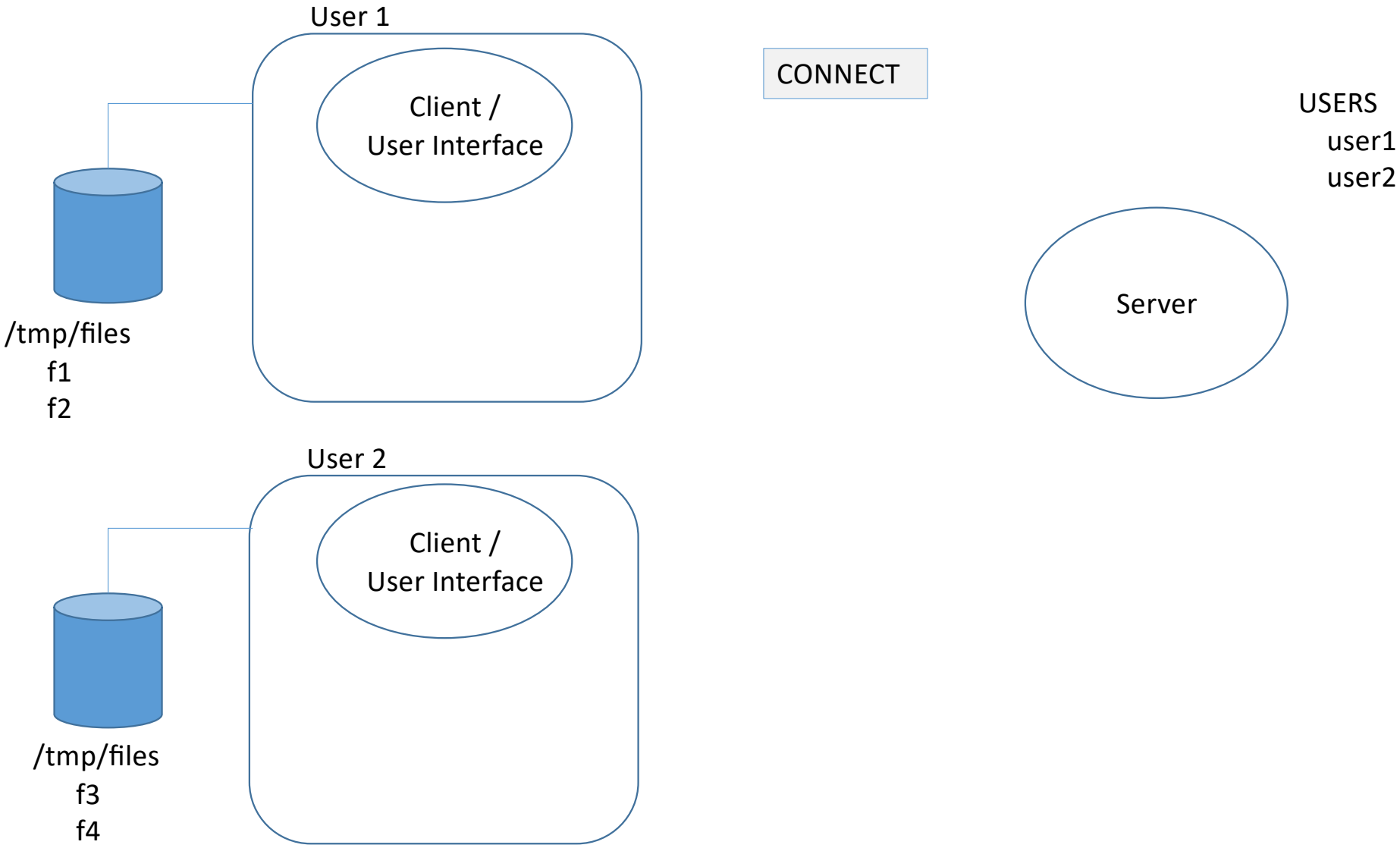

User 1
/tmp/files
f1
f2
Server
User 2
/tmp/files
f3
f4

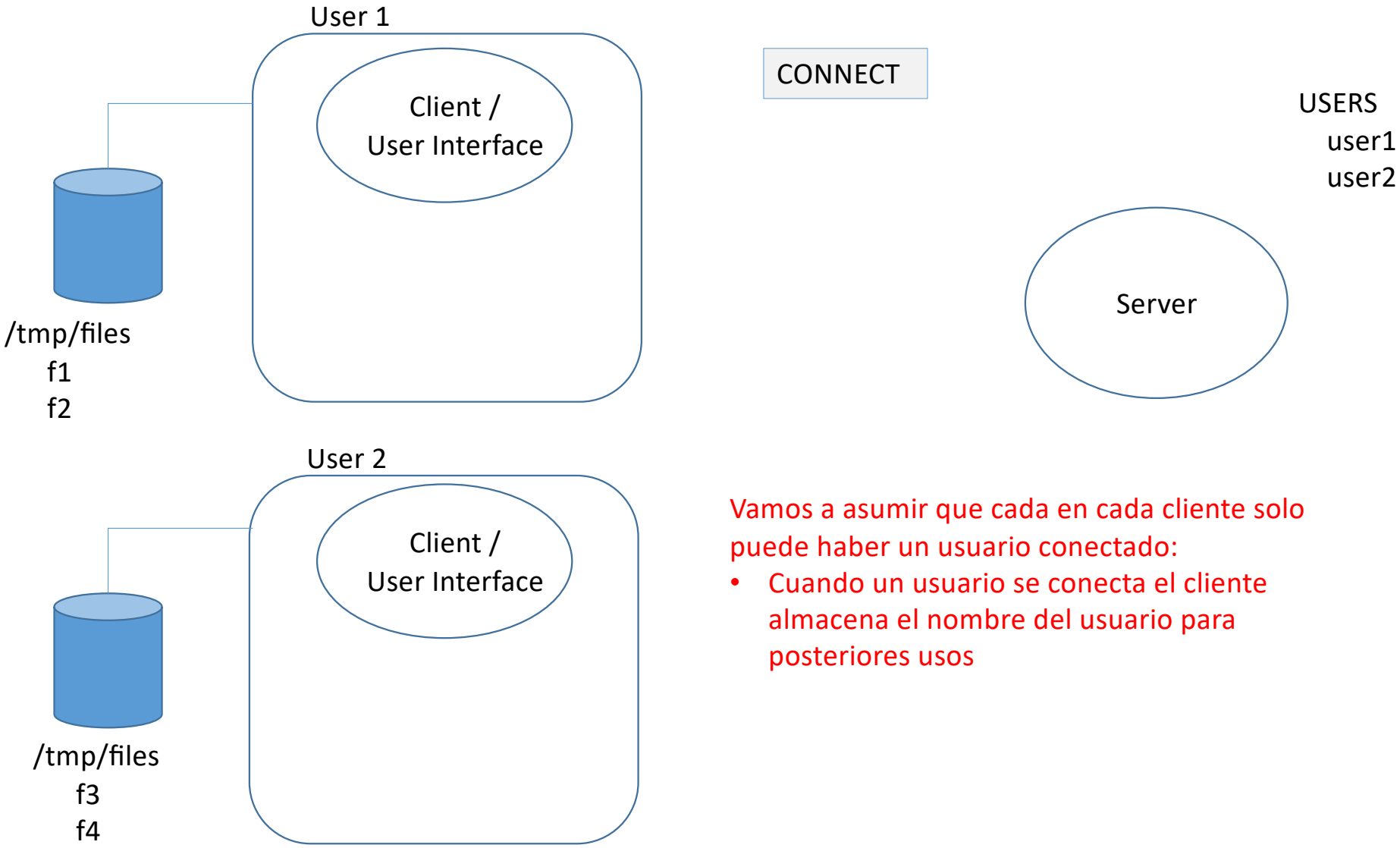

User 1
Client /
User Interface
REGISTER
Server
/tmp/files
f1
f2
User 2
Client /
User Interface
/tmp/files
f3
f4


User 1
Client /
User Interface
REGISTER user1
Server
/tmp/files
f1
f2
User 2
Client /
User Interface
/tmp/files
f3
f4


User 1
Client /
User Interface
/tmp/files
f1
f2
REGISTER user1
USERS
user1
Server
User 2
Client /
User Interface
/tmp/files
f3
f4

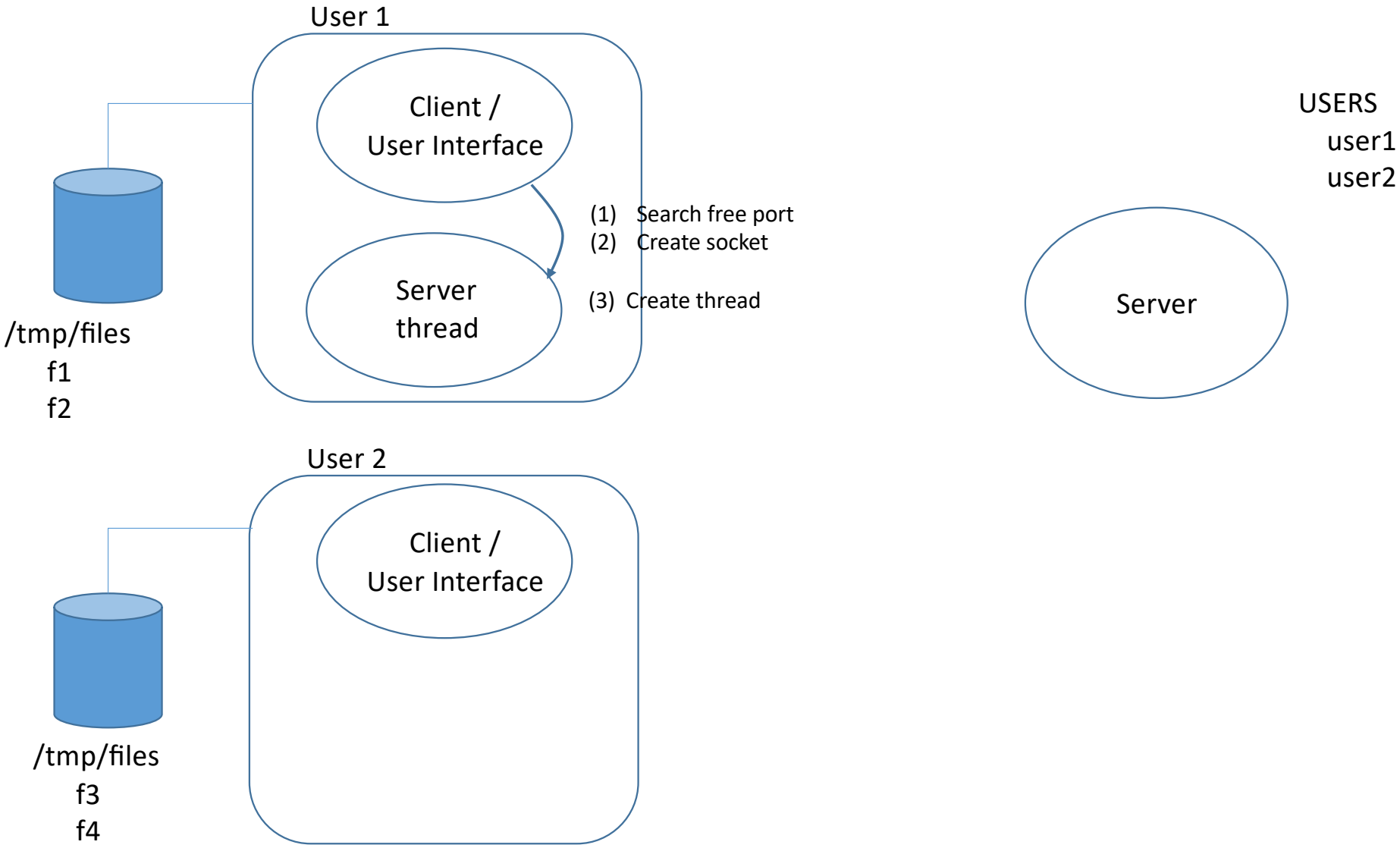

User 1
Client /
User Interface
/tmp/files
f1
f2
REGISTER user1
USERS
user1
user2
Server
User 2
Client /
User Interface
REGISTER user2
/tmp/files
f3
f4

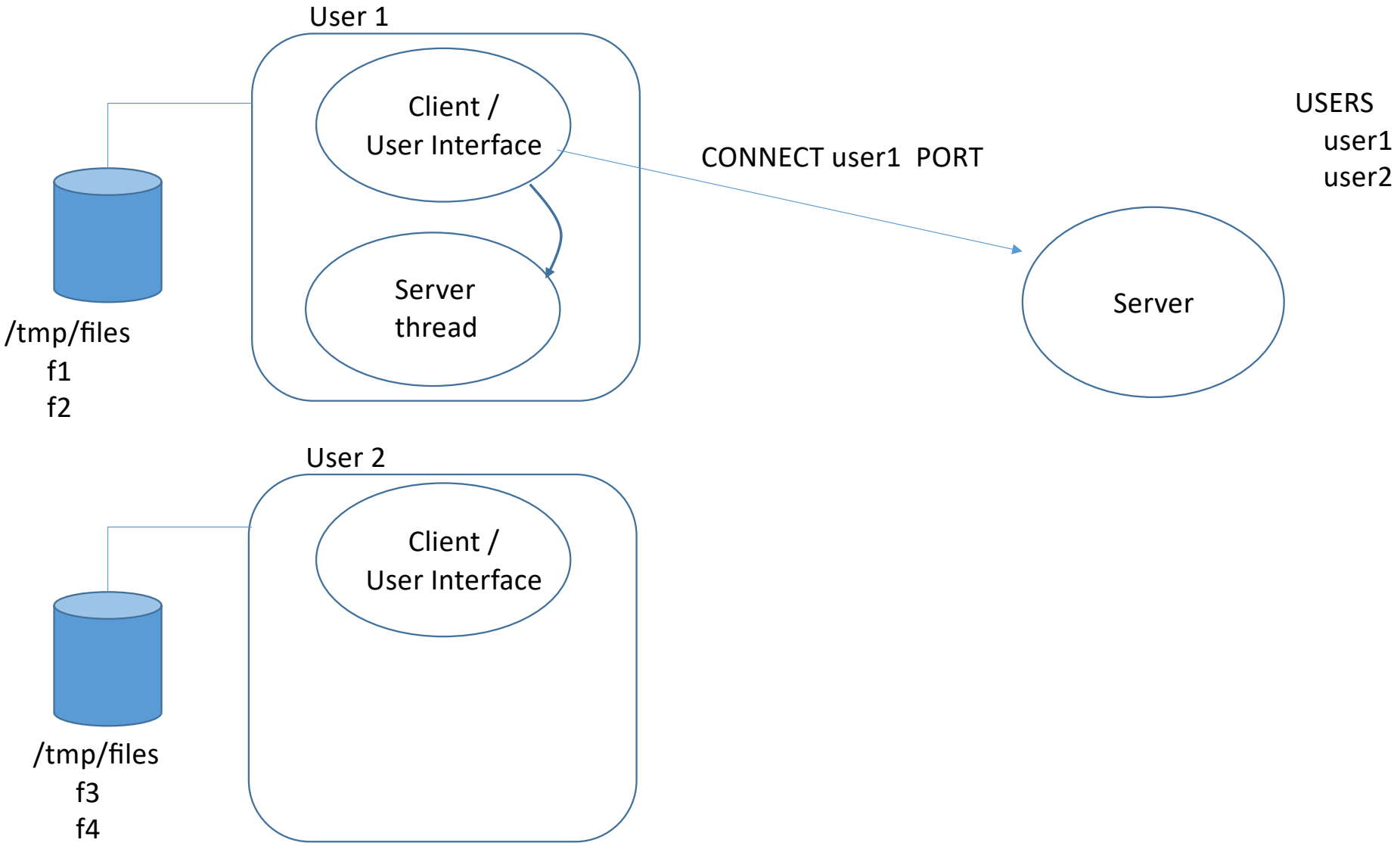

User 1
Client /
User Interface
/tmp/files
f1
f2
CONNECT
USERS
user1
user2
Server
User 2
Client /
User Interface
/tmp/files
f3
f4

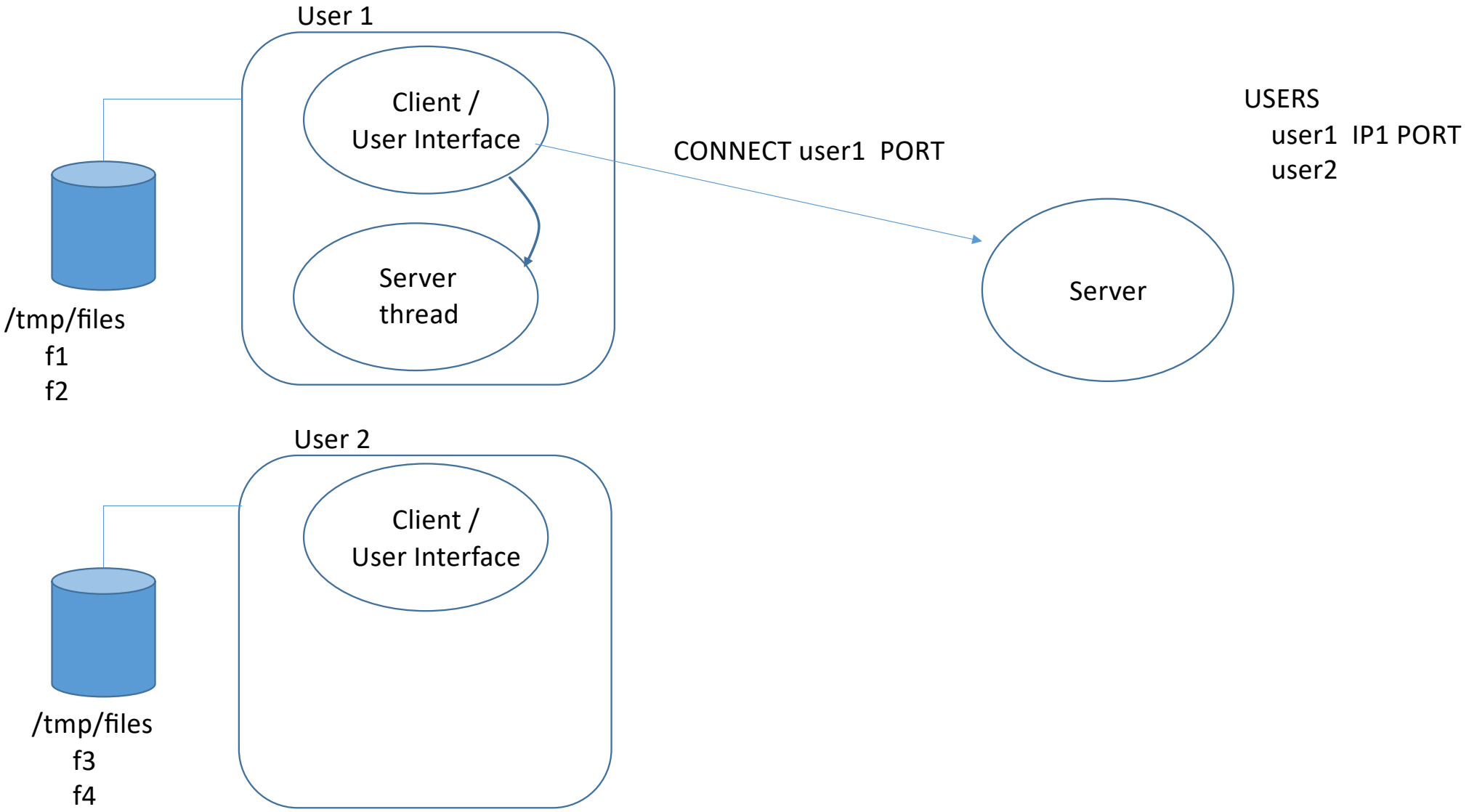


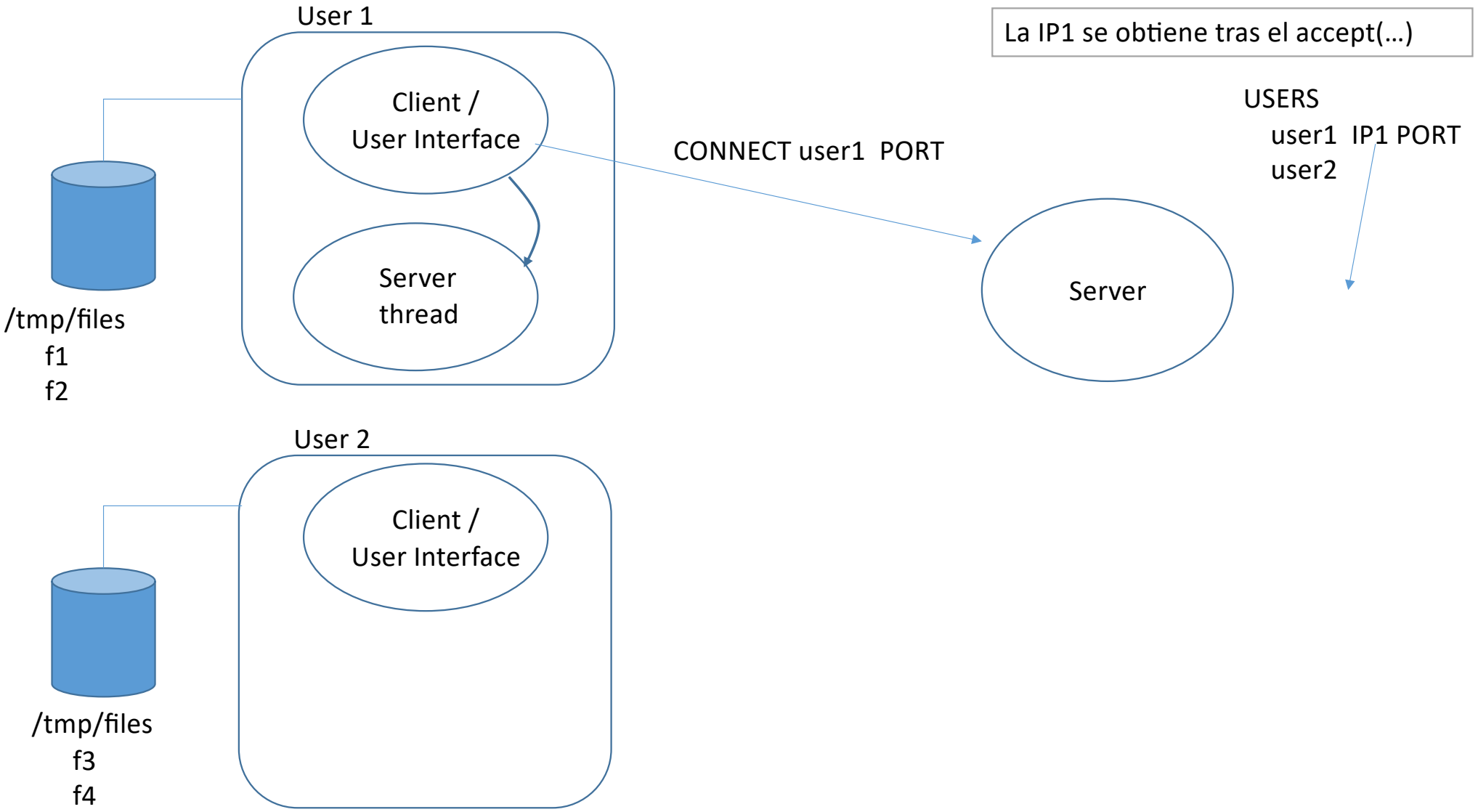
Vamos a asumir que cada en cada cliente solo puede haber un usuario conectado:

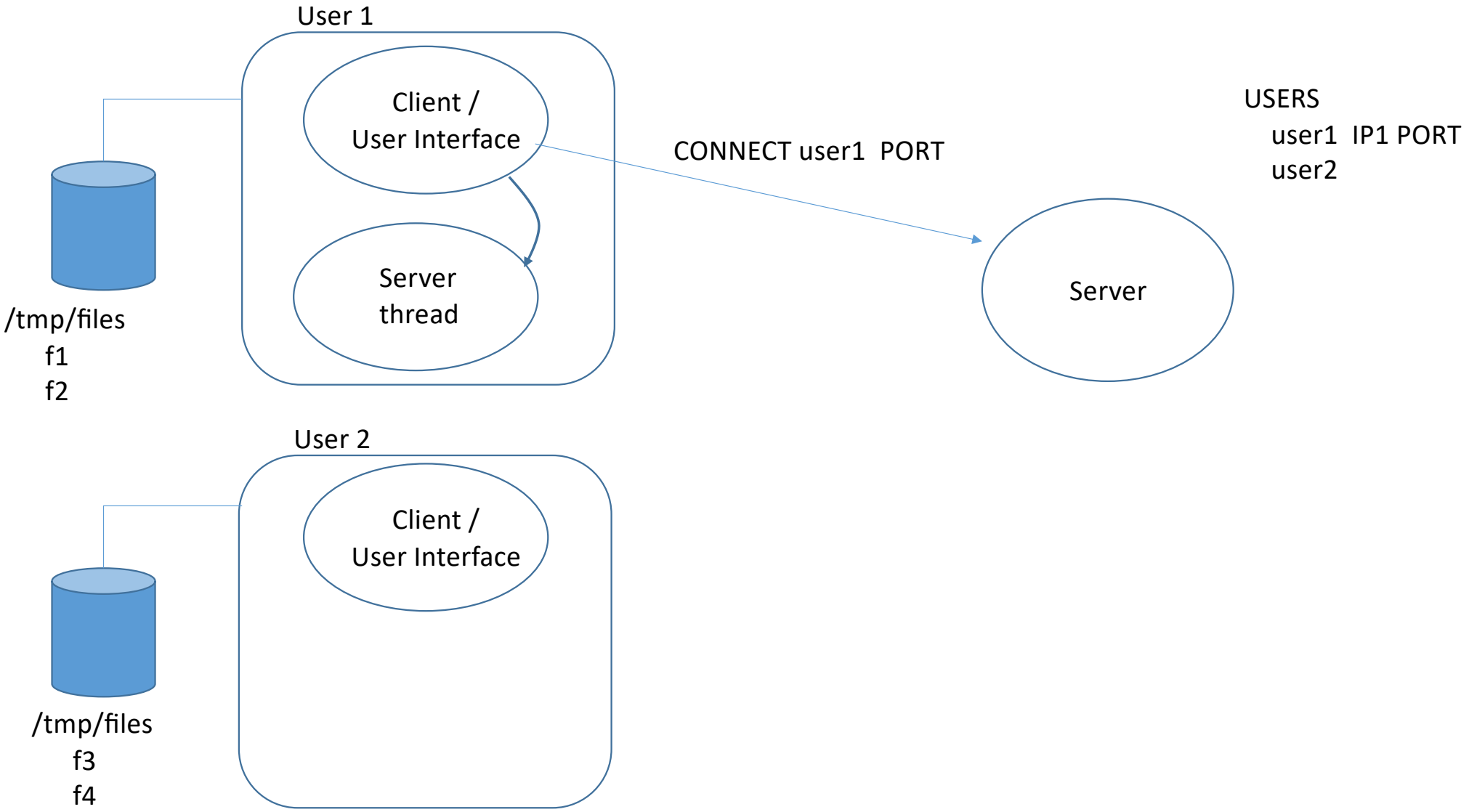
- Cuando un usuario se conecta el cliente almacena el nombre del usuario para posteriores usos

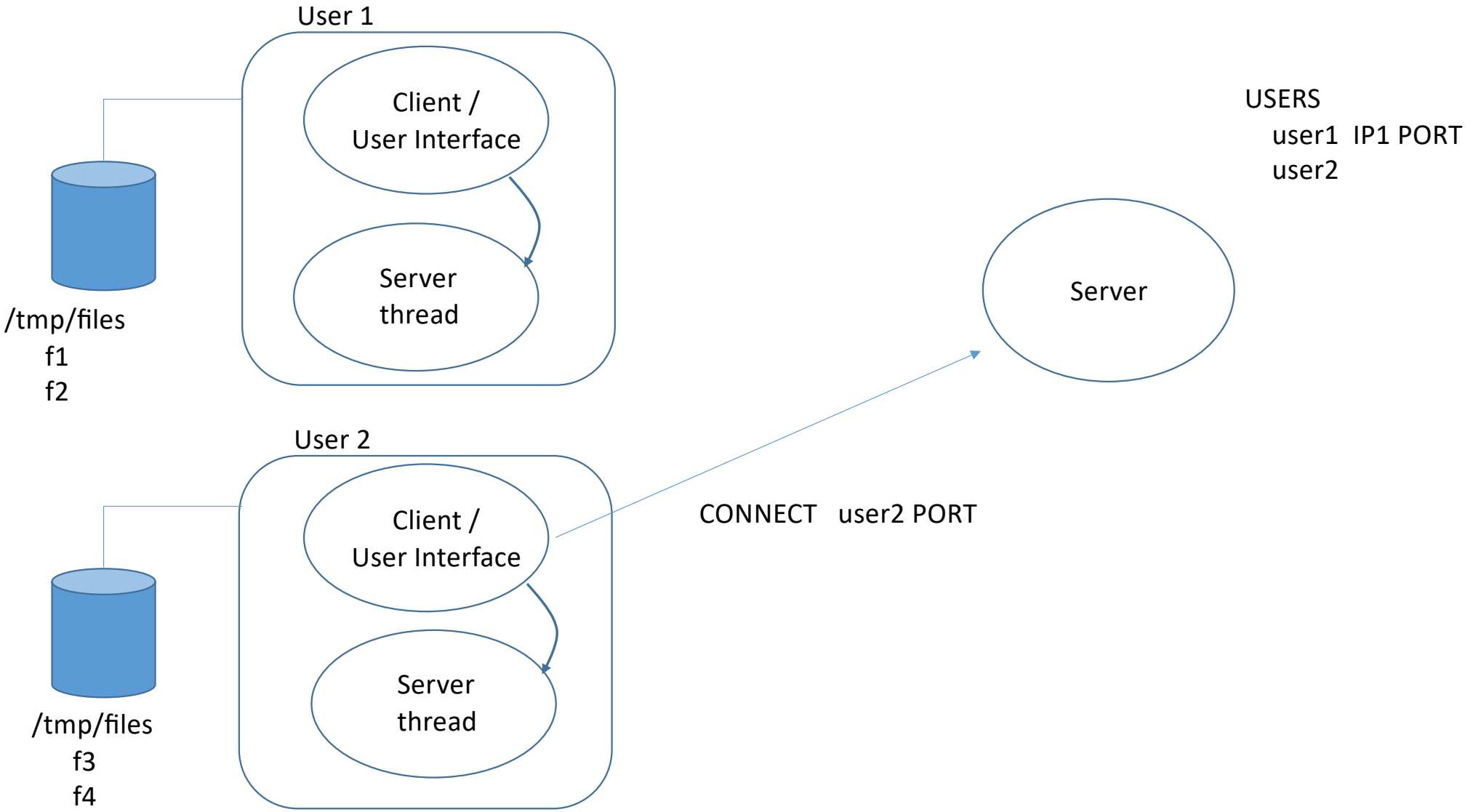

User 1
Client /
User Interface
(1) Search free port
(2) Create socket
Server
thread
(3) Create thread
/tmp/files
f1
f2
USERS
user1
user2
Server
User 2
Client /
User Interface
/tmp/files
f3
f4

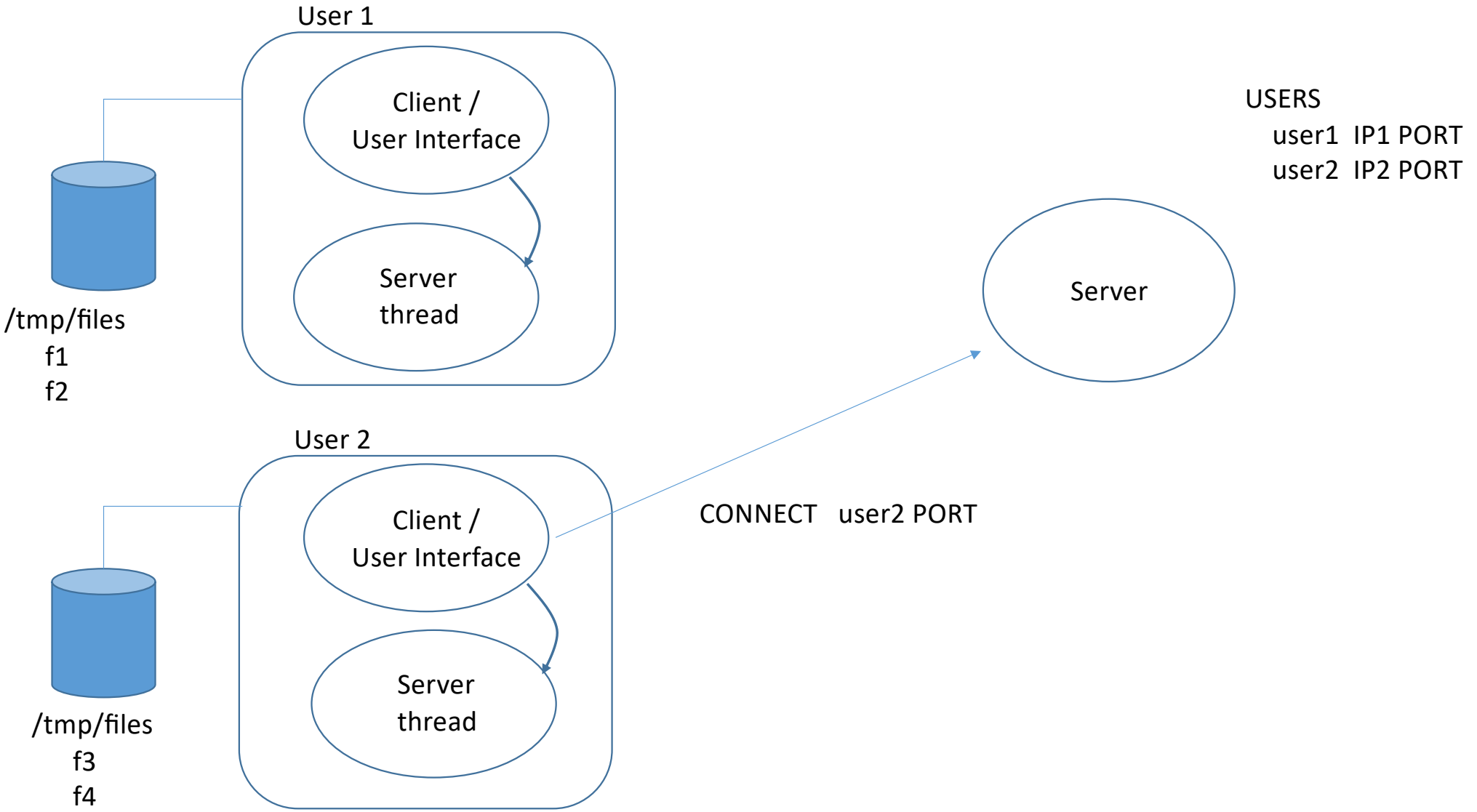

User 1
Client /
User Interface
Server
thread
/tmp/files
f1
f2
CONNECT user1 PORT
Server
USERS
user1
user2
User 2
Client /
User Interface
/tmp/files
f3
f4

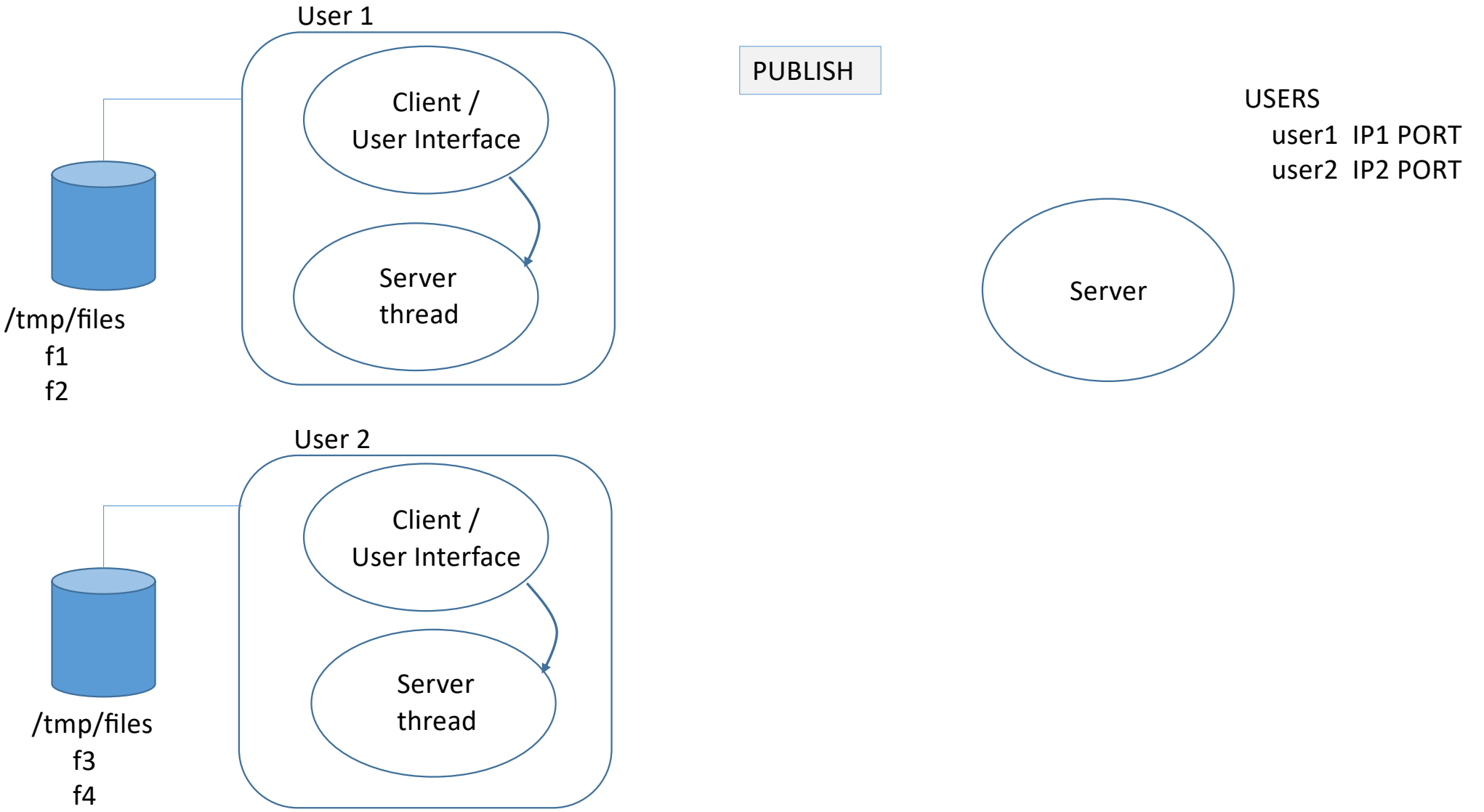

User 1
Client /
User Interface
Server
thread
/tmp/files
f1
f2
CONNECT user1 PORT
Server
USERS
user1 IP1 PORT
user2
User 2
Client /
User Interface
/tmp/files
f3
f4

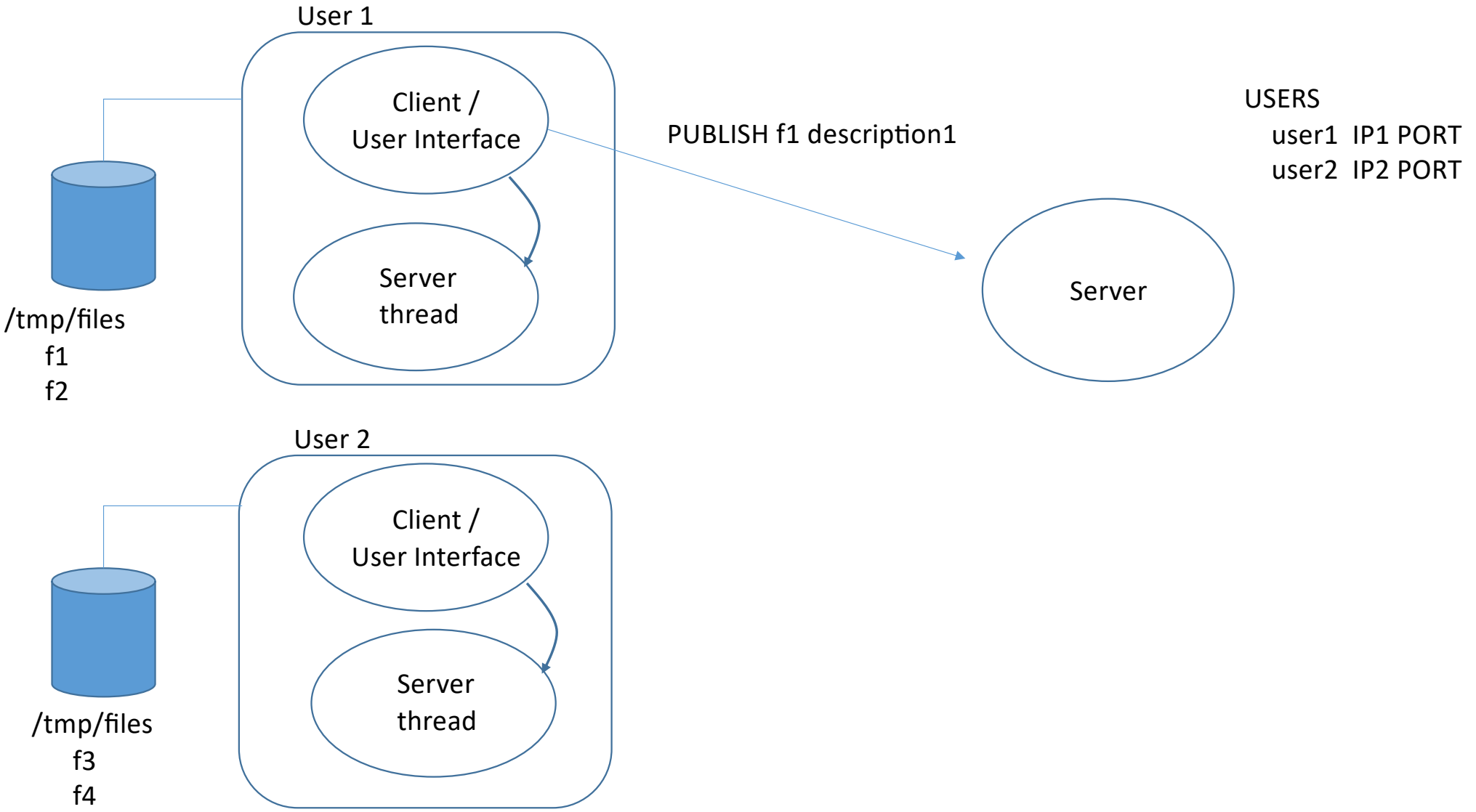

User 1
Client /
User Interface
Server
thread
/tmp/files
f1
f2
CONNECT user1 PORT
La IP1 se obtiene tras el accept(…)
USERS
user1 IP1 PORT
user2
Server
User 2
Client /
User Interface
/tmp/files
f3
f4

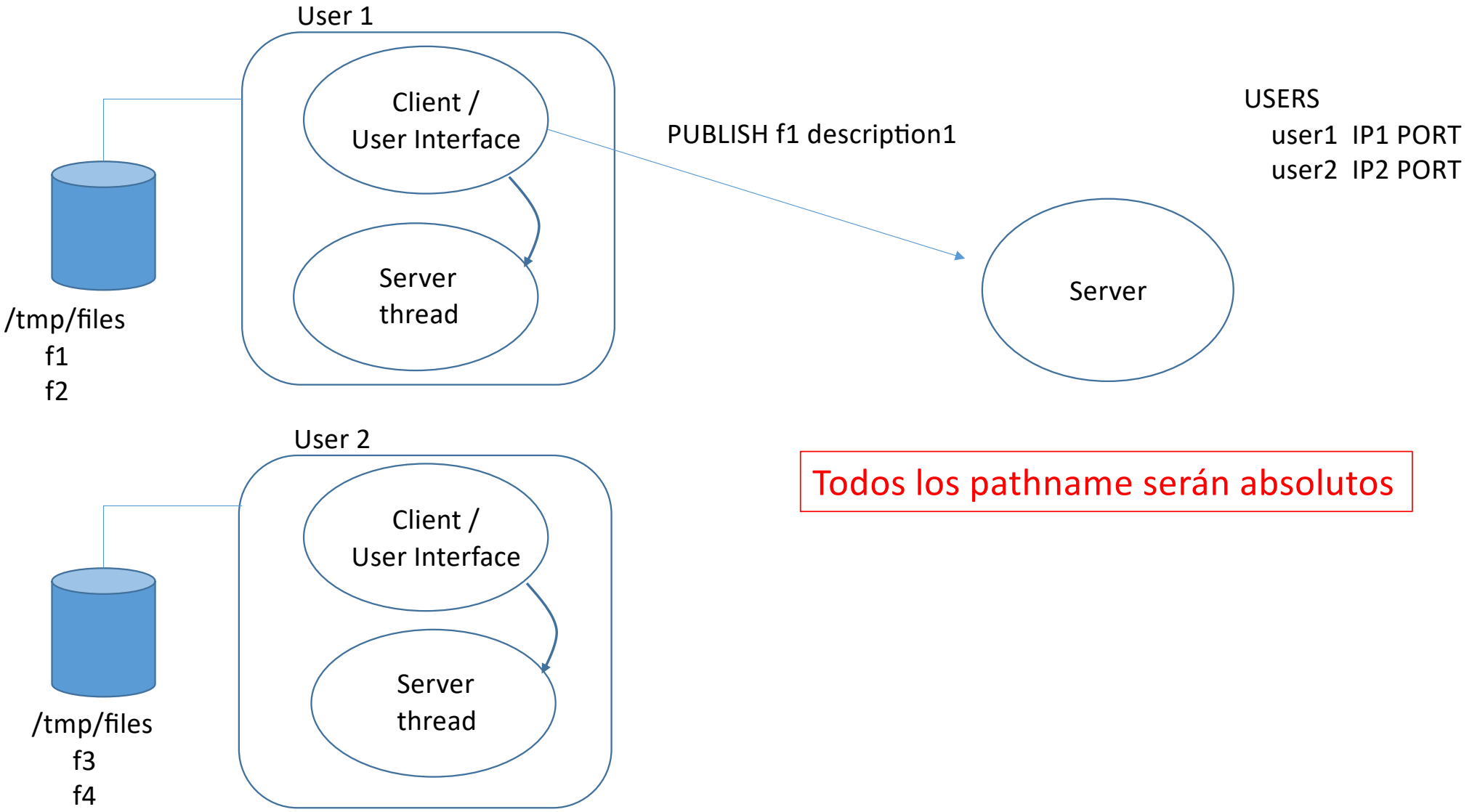

User 1
Client /
User Interface
Server
thread
/tmp/files
f1
f2
CONNECT user1 PORT
Server
USERS
user1 IP1 PORT
user2
User 2
Client /
User Interface
/tmp/files
f3
f4

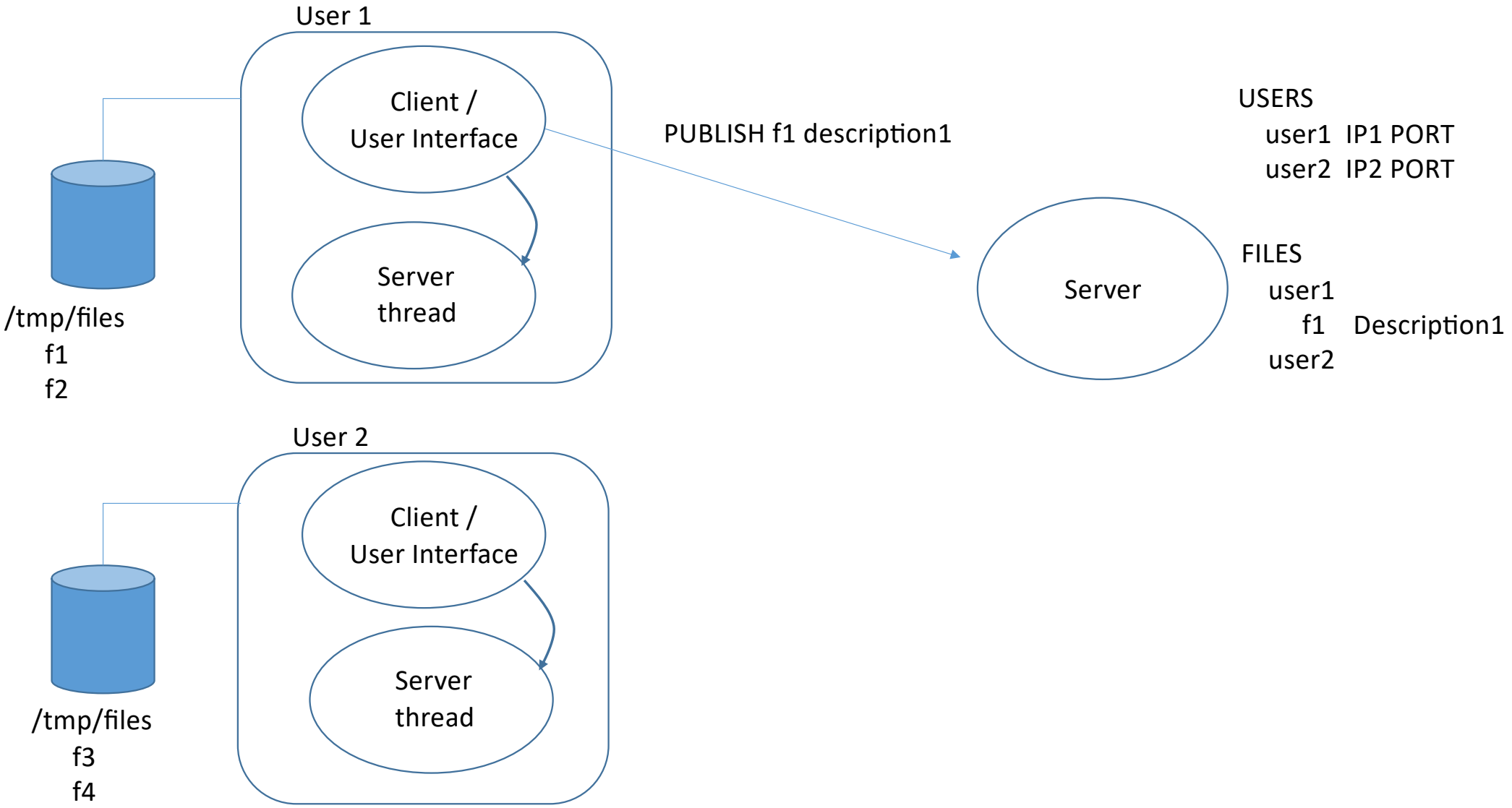

User 1
Client /
User Interface
Server
thread
/tmp/files
f1
f2
User 2
Client /
User Interface
Server
thread
/tmp/files
f3
f4
Server
USERS
user1 IP1 PORT
user2
CONNECT user2 PORT

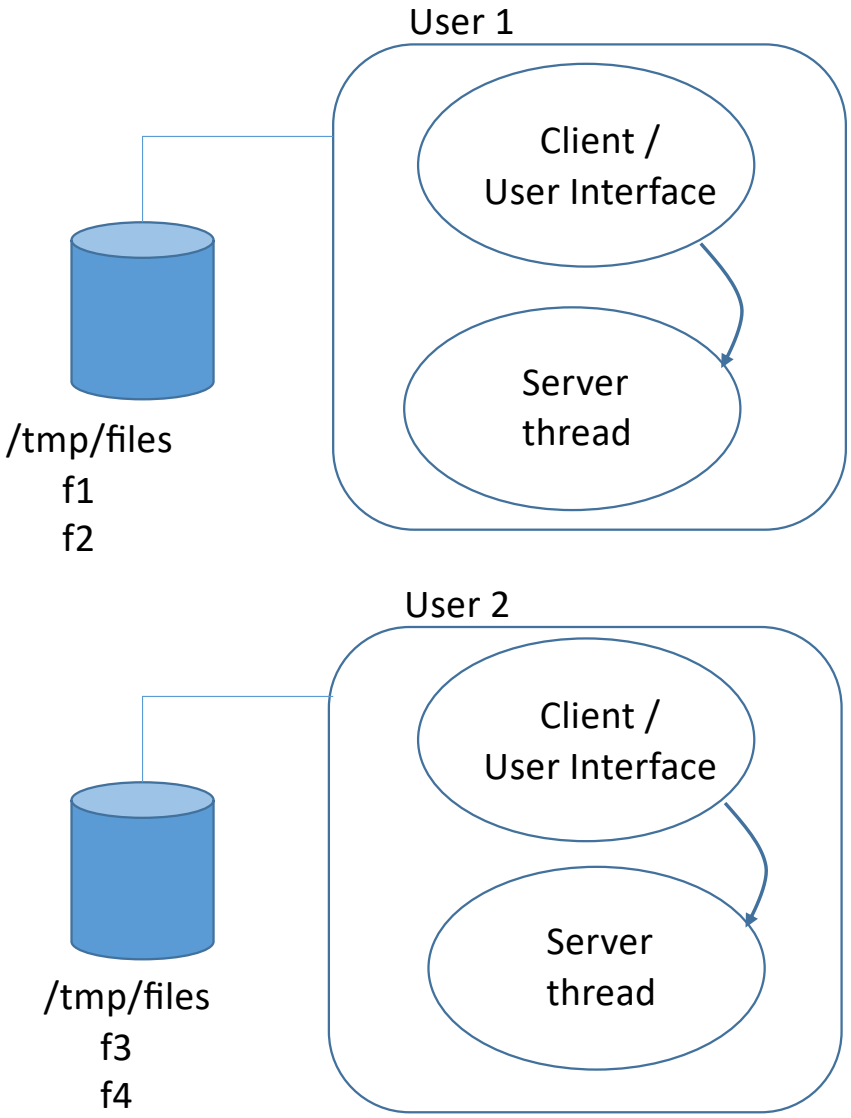

User 1
Client /
User Interface
Server
thread
/tmp/files
f1
f2
User 2
Client /
User Interface
Server
thread
/tmp/files
f3
f4
Server
USERS
user1 IP1 PORT
user2 IP2 PORT
CONNECT user2 PORT


User 1
Client /
User Interface
Server
thread
/tmp/files
f1
f2
PUBLISH
USERS
user1 IP1 PORT
user2 IP2 PORT
Server
User 2
Client /
User Interface
Server
thread
/tmp/files
f3
f4


User 1
Client /
User Interface
Server
thread
/tmp/files
f1
f2
PUBLISH f1 description1
Server
USERS
user1 IP1 PORT
user2 IP2 PORT
User 2
Client /
User Interface
Server
thread
/tmp/files
f3
f4


User 1
Client /
User Interface
Server
thread
/tmp/files
f1
f2
PUBLISH f1 description1
Server
USERS
user1 IP1 PORT
user2 IP2 PORT
User 2
Client /
User Interface
Server
thread
/tmp/files
f3
f4
Todos los pathname serán absolutos


User 1
Client /
User Interface
Server
thread
/tmp/files
f1
f2
PUBLISH f1 description1
Server
USERS
user1 IP1 PORT
user2 IP2 PORT
FILES
user1
f1 Description1
user2
User 2
Client /
User Interface
Server
thread
/tmp/files
f3
f4



Server

USERS

user1 IP1 PORT

user2 IP2 PORT

FILES

user1

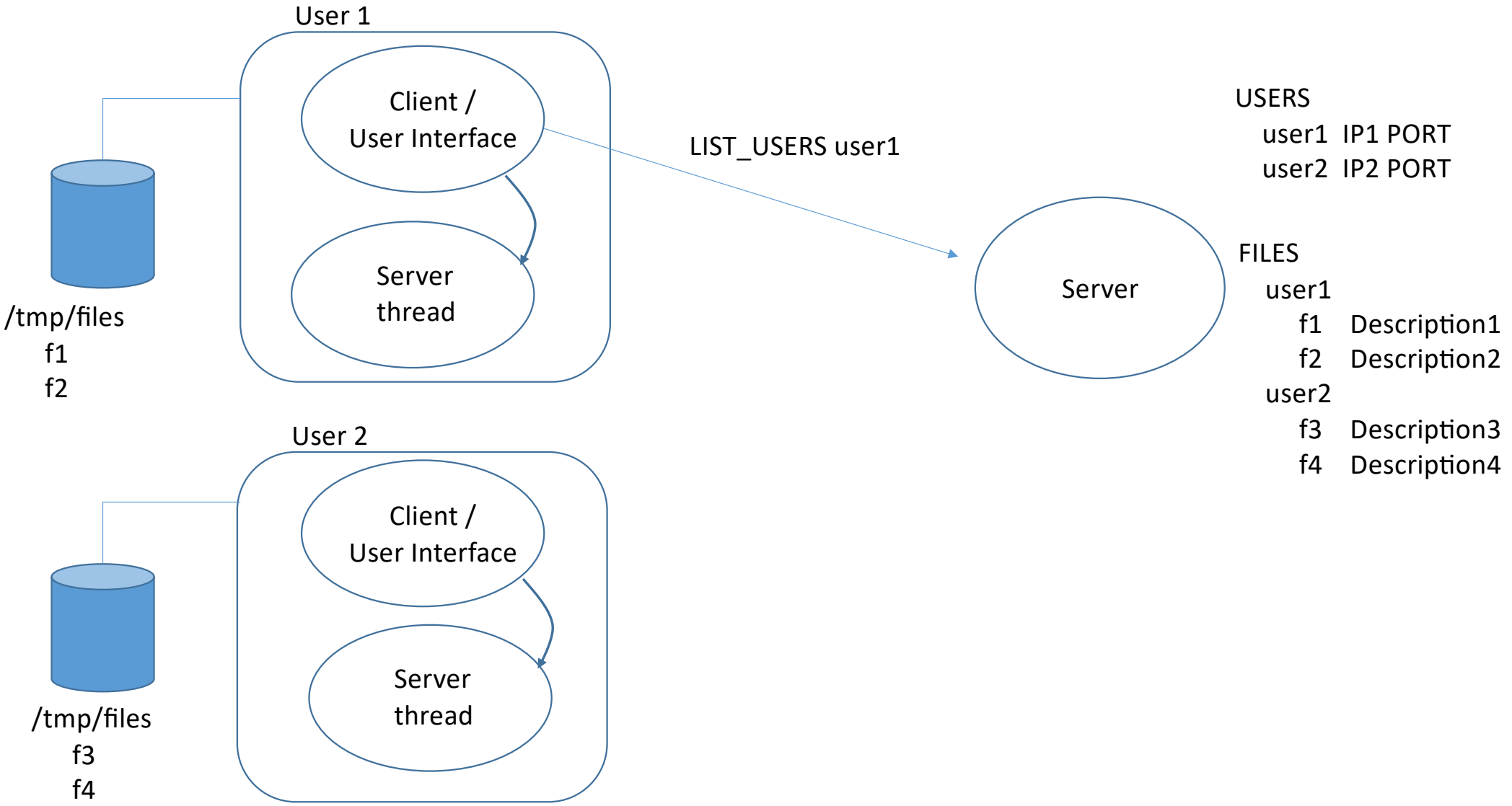
f1 Description1

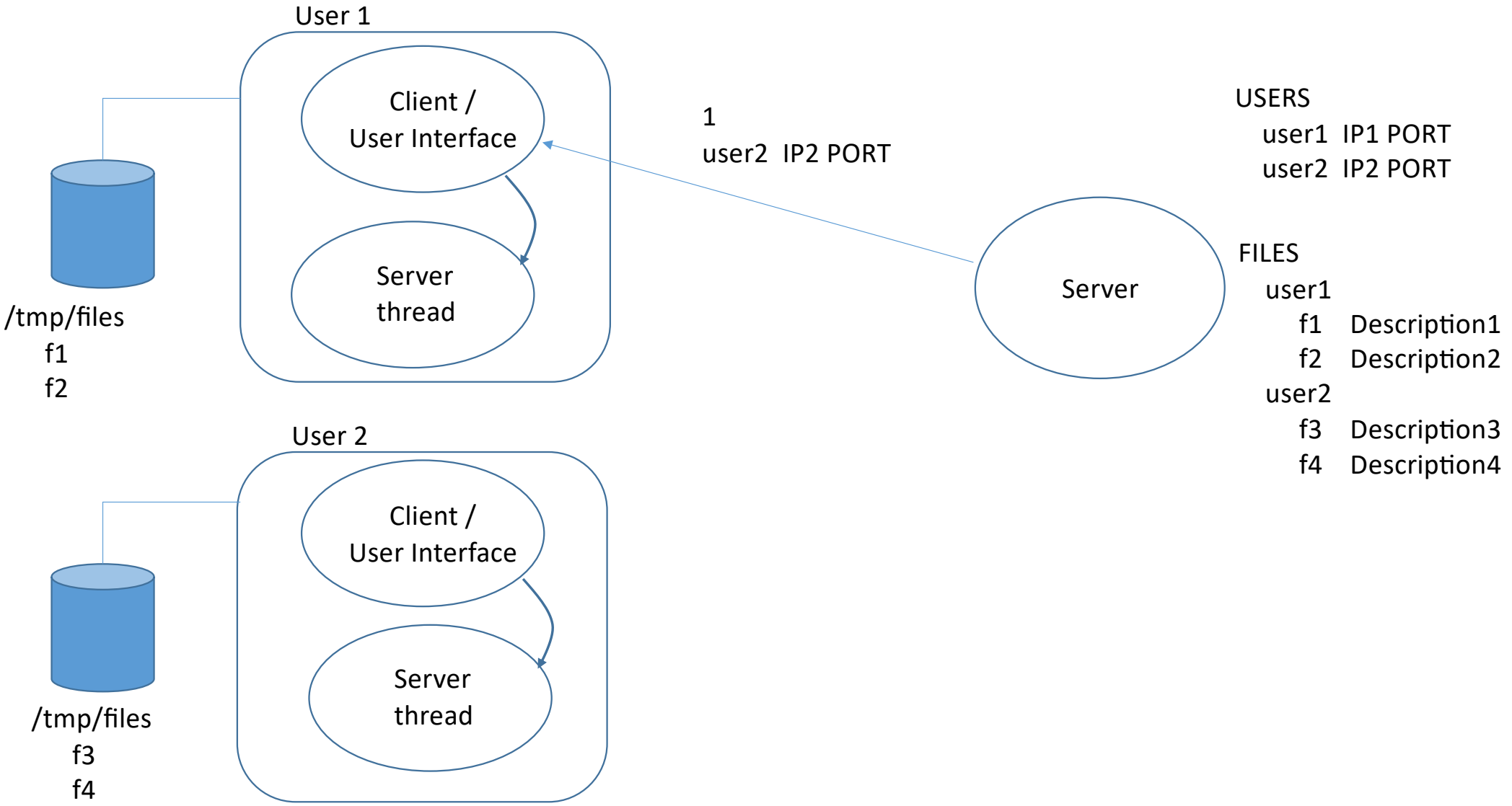
f2 Description2

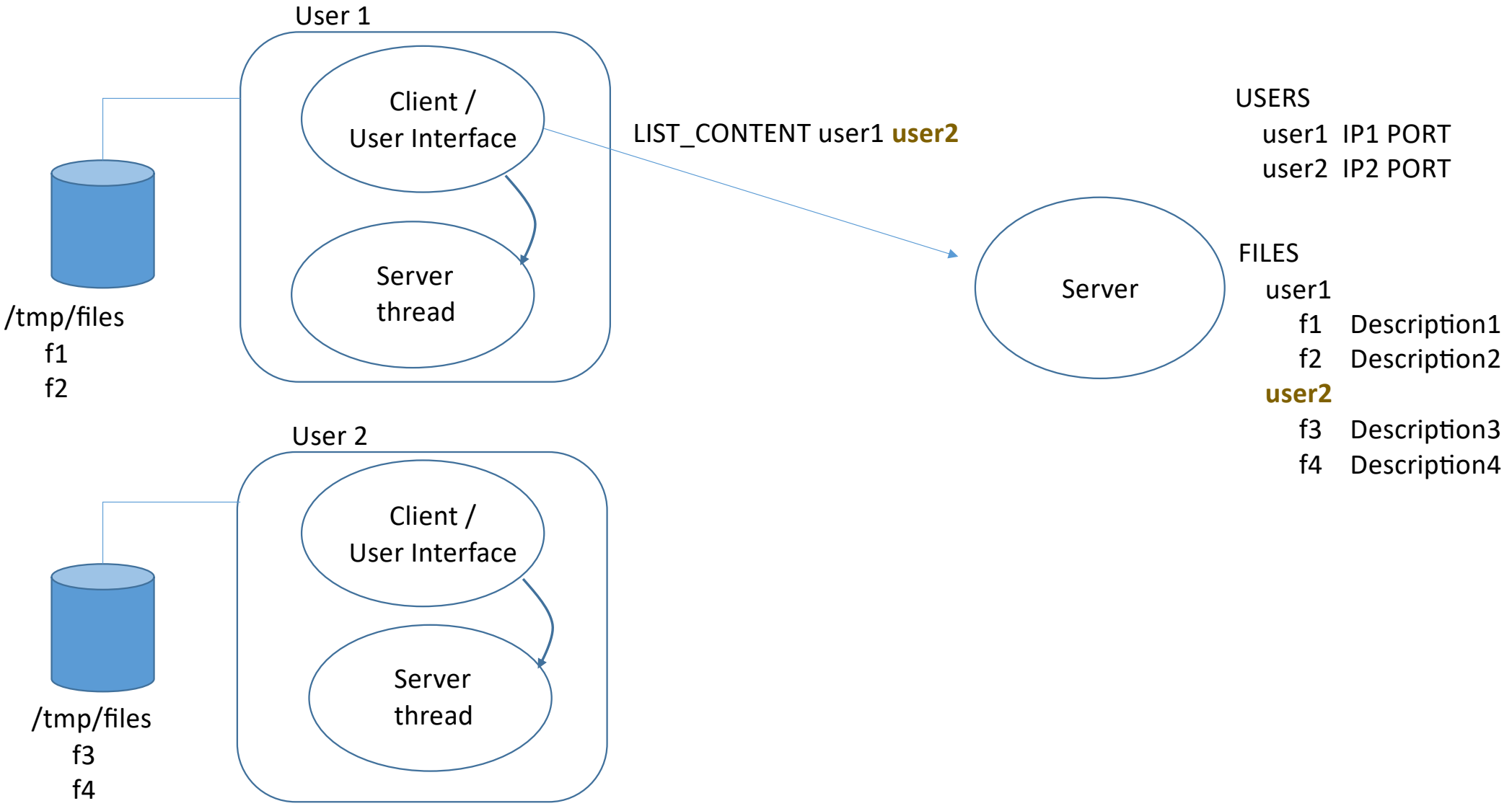
user2

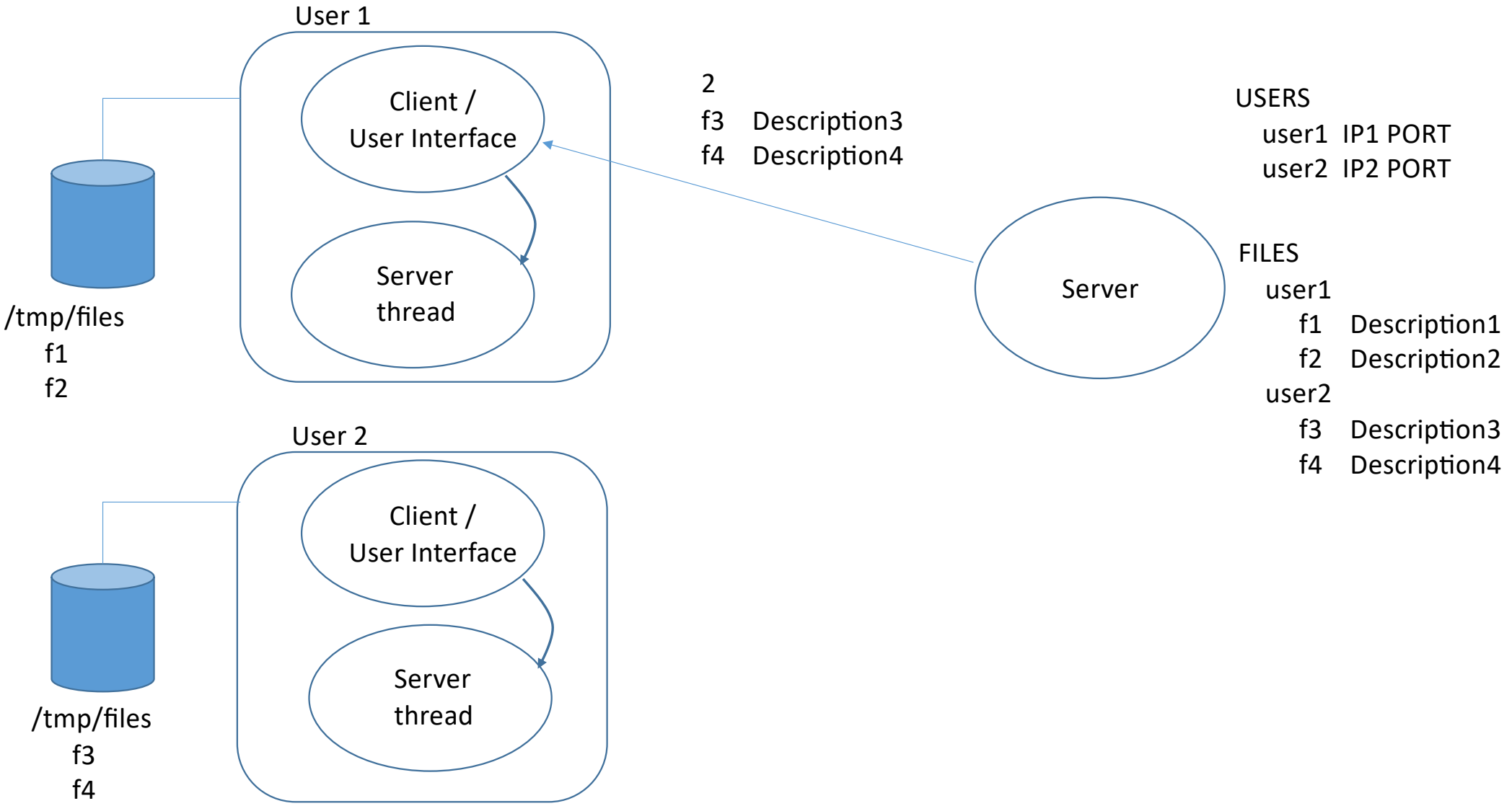
f3 Description3

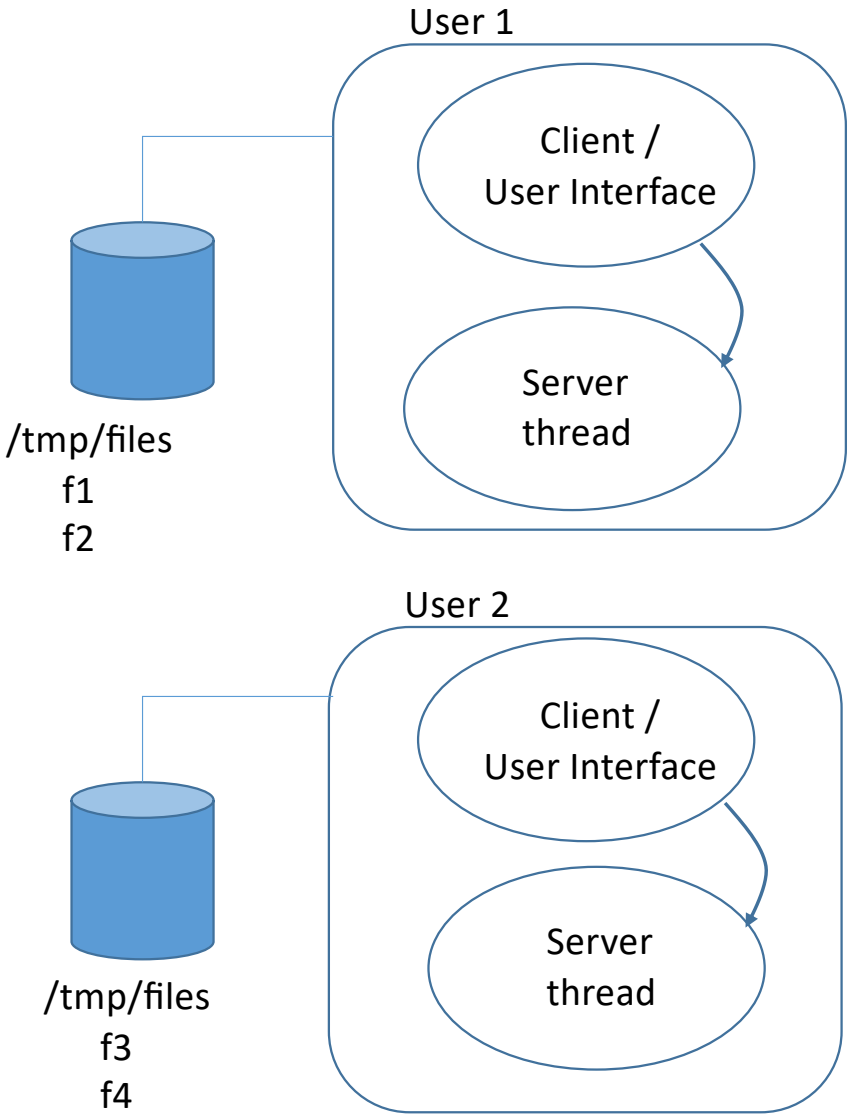
f4 Description4


User 1
Client /
User Interface
Server
thread
/tmp/files
f1
f2
LIST_USERS user1
Server
USERS
user1 IP1 PORT
user2 IP2 PORT
FILES
user1
f1 Description1
f2 Description2
user2
f3 Description3
f4 Description4
User 2
Client /
User Interface
Server
thread
/tmp/files
f3
f4

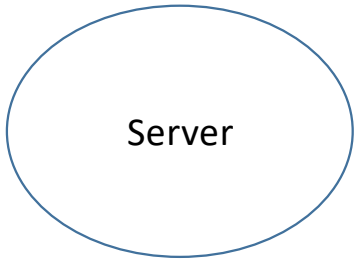

User 1
Client /
User Interface
Server
thread
/tmp/files
f1
f2
1
user2 IP2 PORT
Server
USERS
user1 IP1 PORT
user2 IP2 PORT
FILES
user1
f1 Description1
f2 Description2
user2
f3 Description3
f4 Description4
User 2
Client /
User Interface
Server
thread
/tmp/files
f3
f4


User 1
Client /
User Interface
Server
thread
/tmp/files
f1
f2
LIST_CONTENT user1 user2
Server
USERS
user1 IP1 PORT
user2 IP2 PORT
FILES
user1
f1 Description1
f2 Description2
user2
f3 Description3
f4 Description4
User 2
Client /
User Interface
Server
thread
/tmp/files
f3
f4


User 1
Client /
User Interface
Server
thread
/tmp/files
f1
f2
2
f3 Description3
f4 Description4
Server
USERS
user1 IP1 PORT
user2 IP2 PORT
FILES
user1
f1 Description1
f2 Description2
user2
f3 Description3
f4 Description4
User 2
Client /
User Interface
Server
thread
/tmp/files
f3
f4



2
f3 Description3
f4 Description4

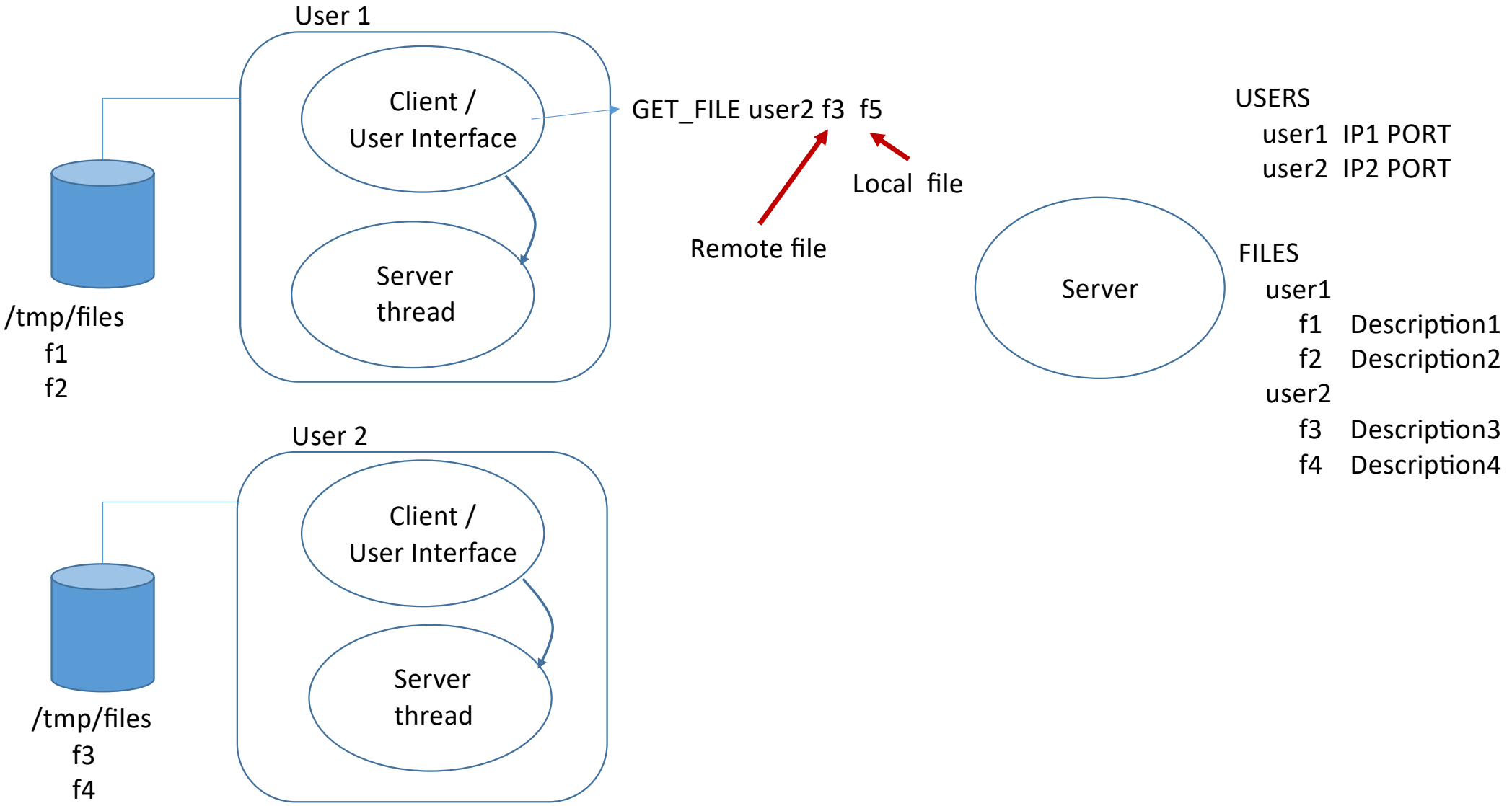


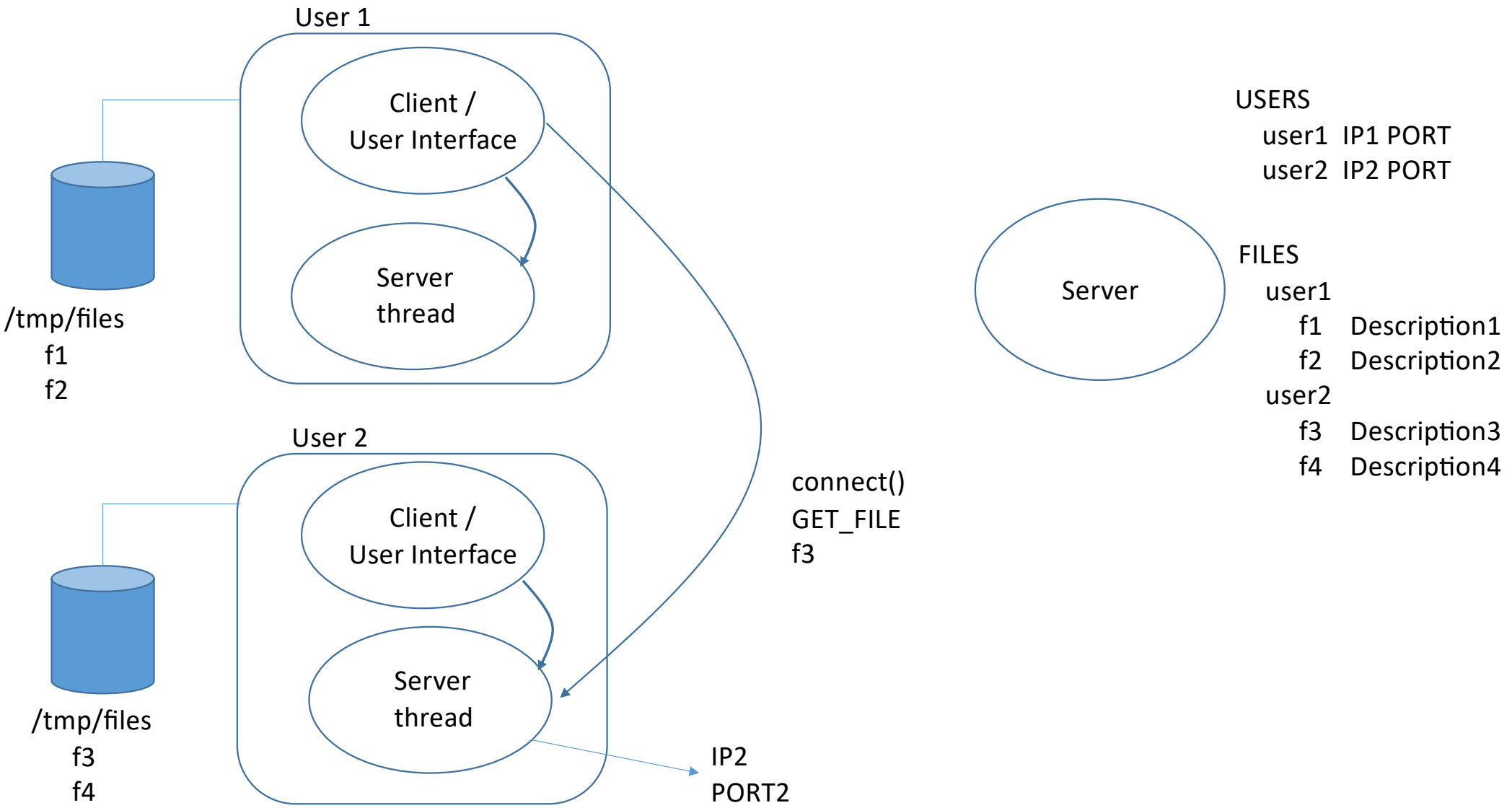
USERS
user1 IP1 PORT
user2 IP2 PORT

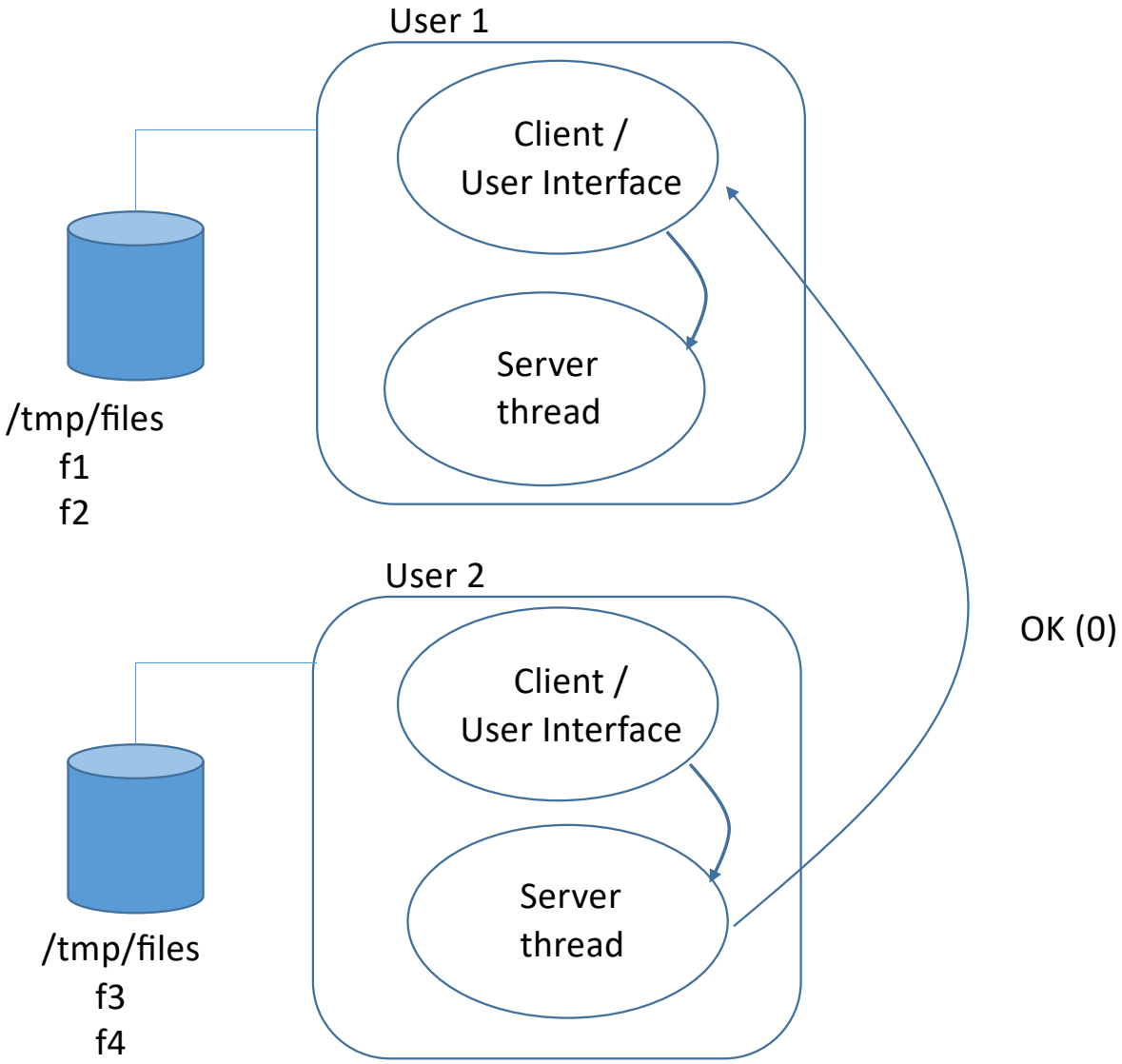
FILES
user1
f1 Description1
f2 Description2
user2
f3 Description3
f4 Description4

El usuario 1 ahora puede estar interesado en descargar a su disco el fichero f3 del usuario/a 2.

Conoce por operación de listar usuarios/as:
IP2 PORT (usuario2)


User 1
Client /
User Interface
Server
thread
/tmp/files
f1
f2
GET_FILE user2 f3 f5
Local file
Remote file
Server
USERS
user1 IP1 PORT
user2 IP2 PORT
FILES
user1
f1 Description1
f2 Description2
user2
f3 Description3
f4 Description4
User 2
Client /
User Interface
Server
thread
/tmp/files
f3
f4


User 1
Client /
User Interface
Server
thread
/tmp/files
f1
f2
User 2
Client /
User Interface
Server
thread
/tmp/files
f3
f4
connect()
GET_FILE
f3
IP2
PORT2
Server
USERS
user1 IP1 PORT
user2 IP2 PORT
FILES
user1
f1 Description1
f2 Description2
user2
f3 Description3
f4 Description4



Server

USERS

user1 IP1 PORT

user2 IP2 PORT

FILES

user1

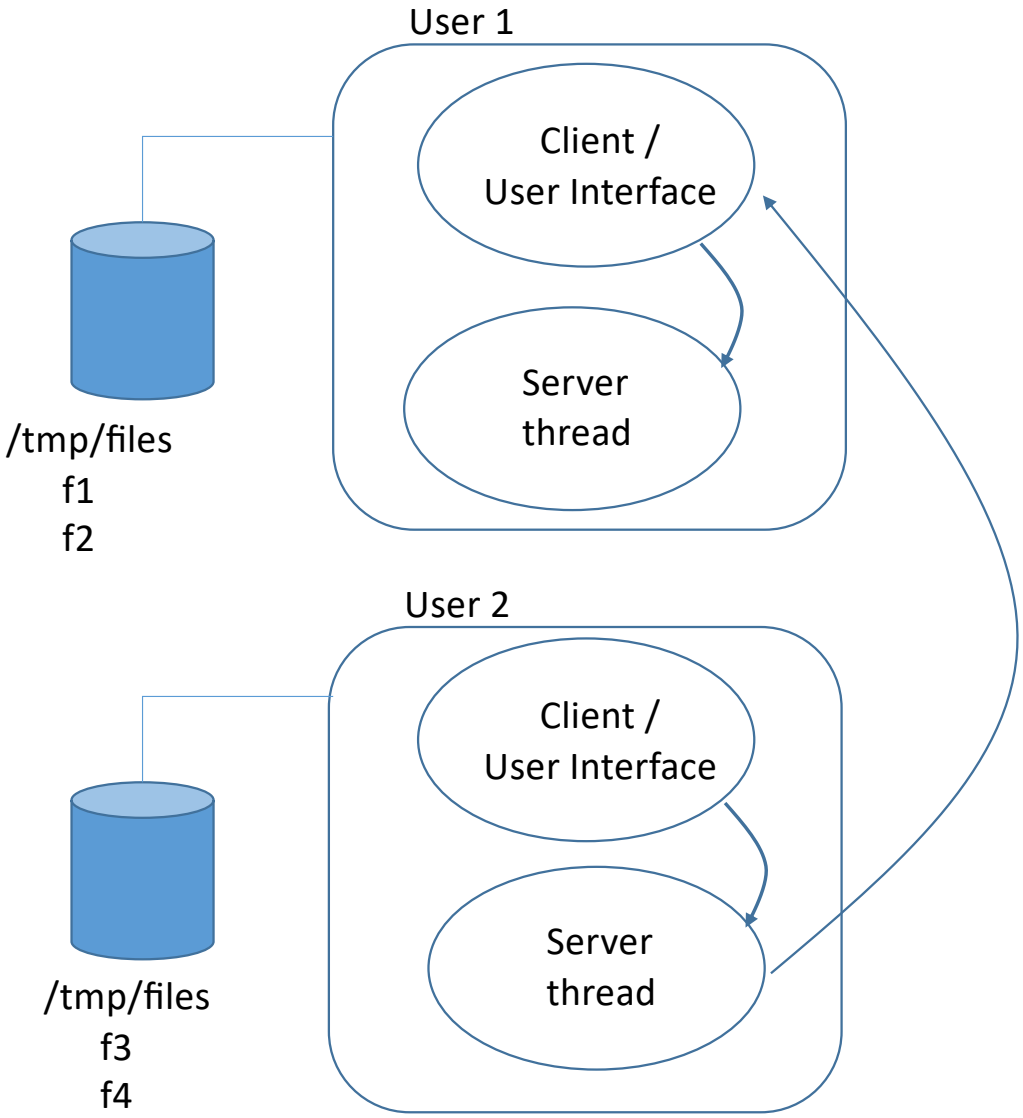
f1 Description1

f2 Description2

user2

f3 Description3

f4 Description4



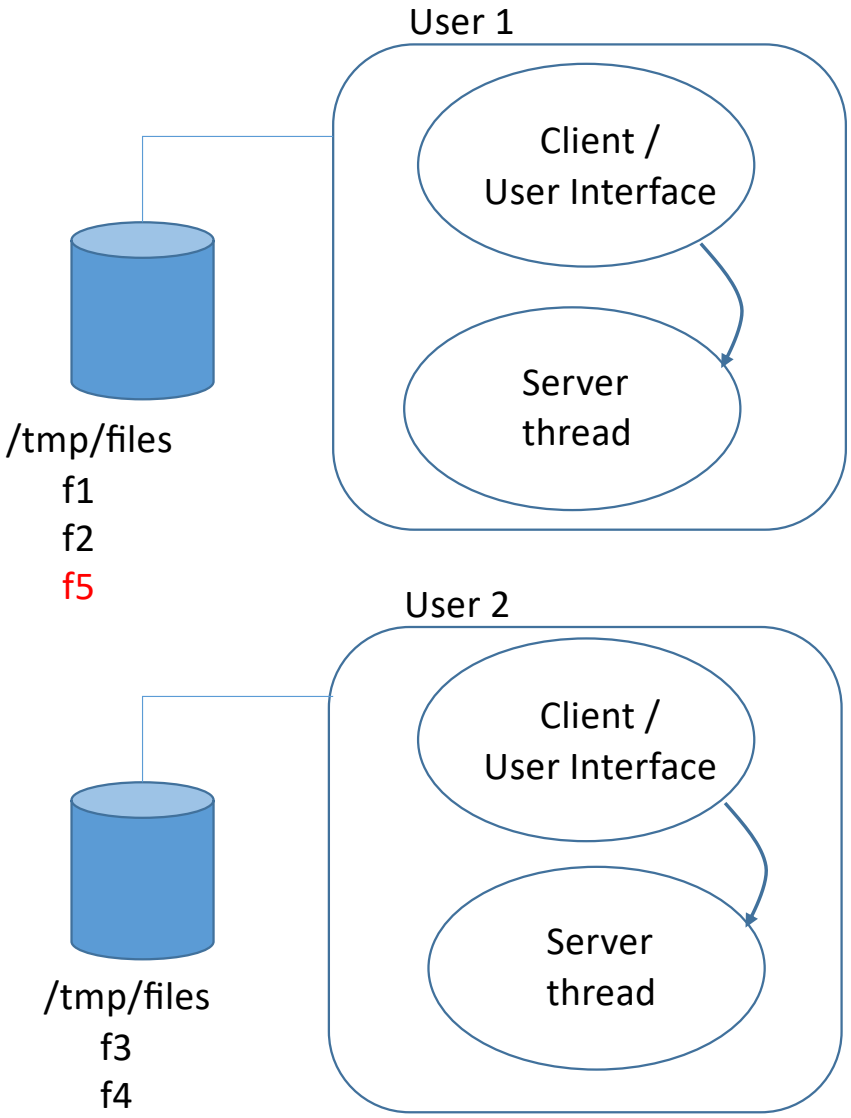
Tamaño en byes (si Ok)

Server

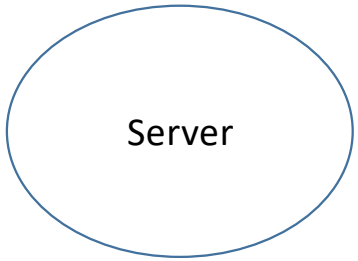
USERS
user1 IP1 PORT
user2 IP2 PORT

FILES
user1
f1 Description1
f2 Description2
user2
f3 Description3
f4 Description4

Todos los pathname serán absolutos

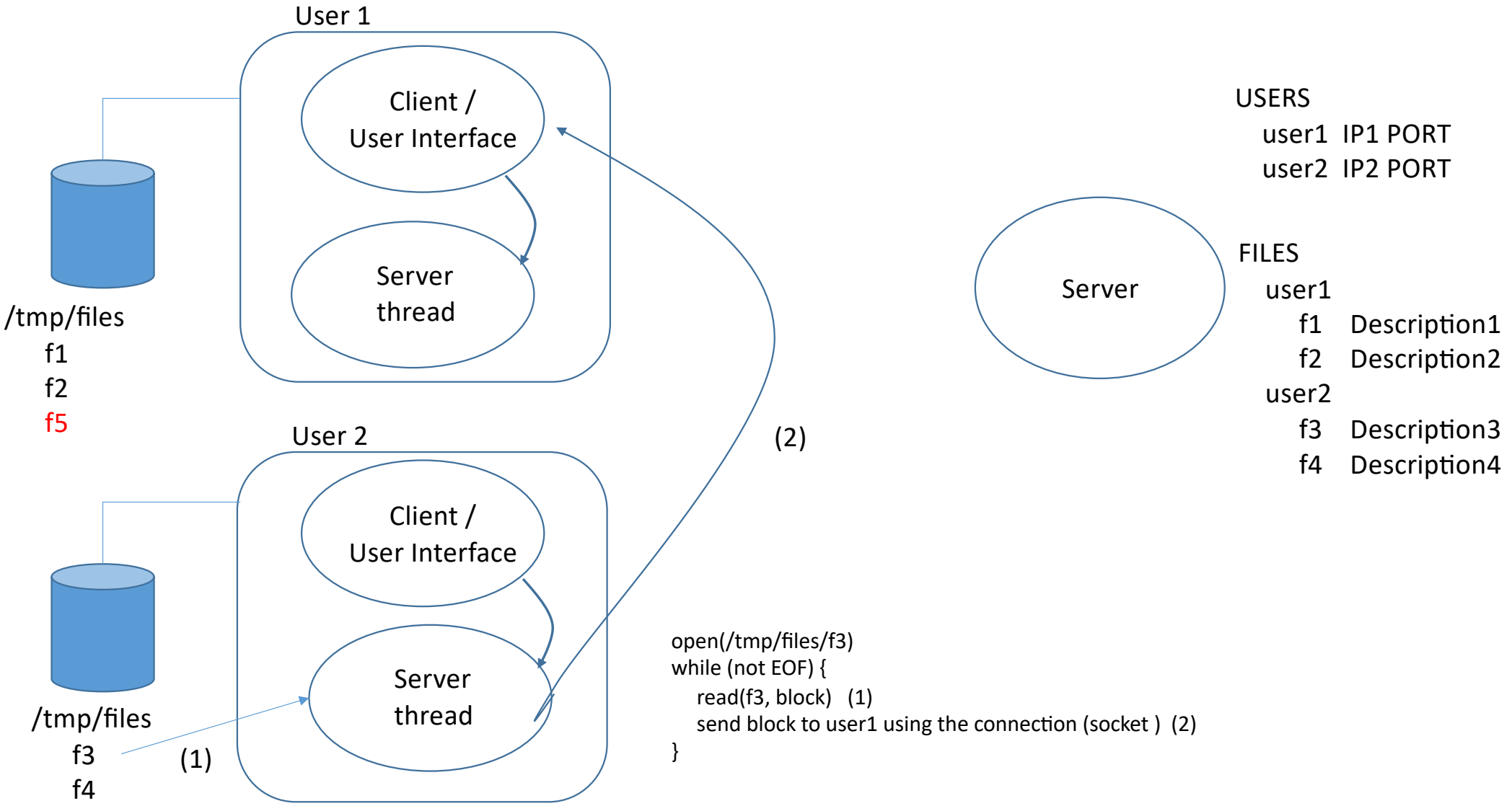


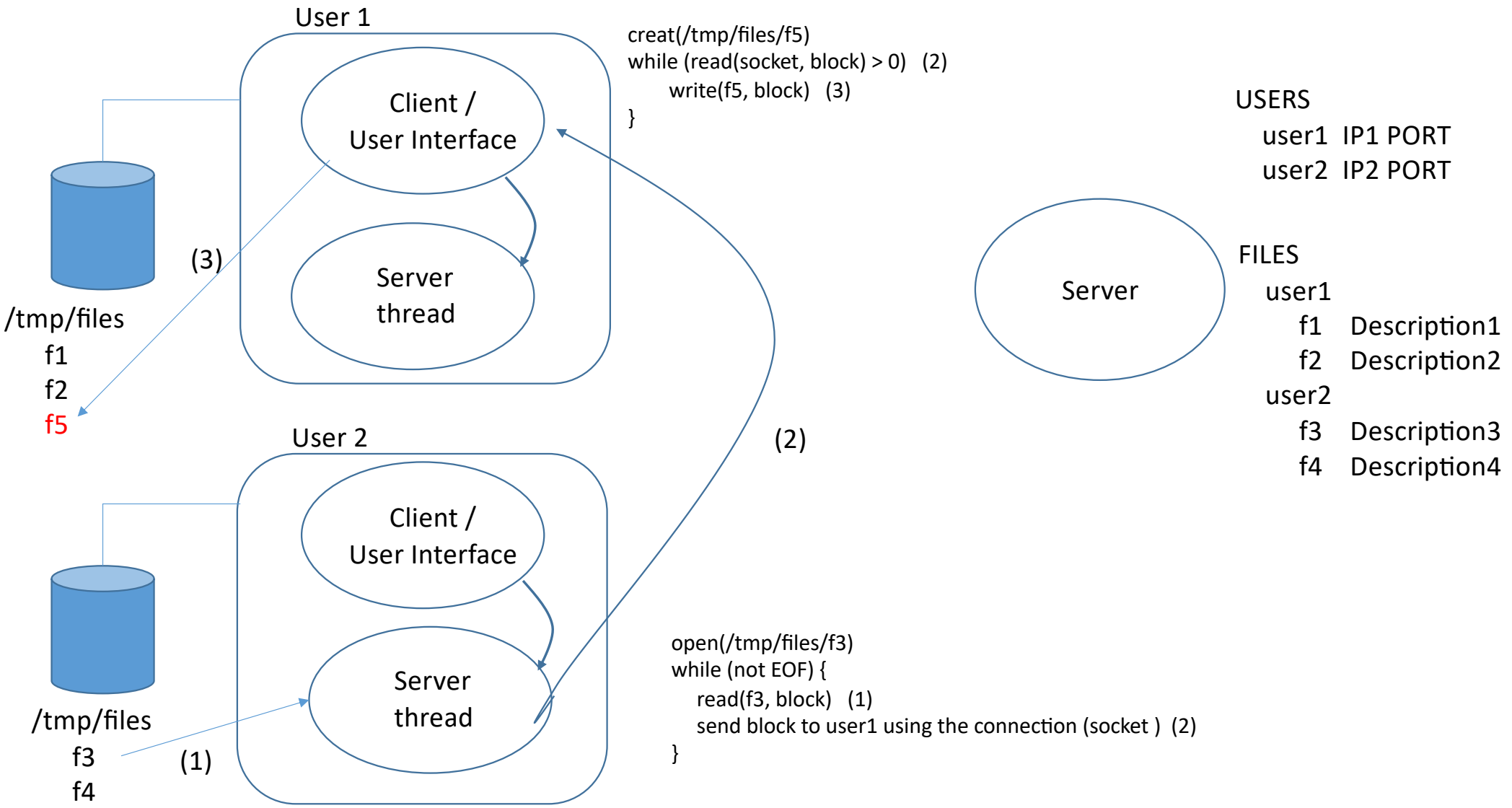
El usuario 1 crea el fichero f5
en su disco local

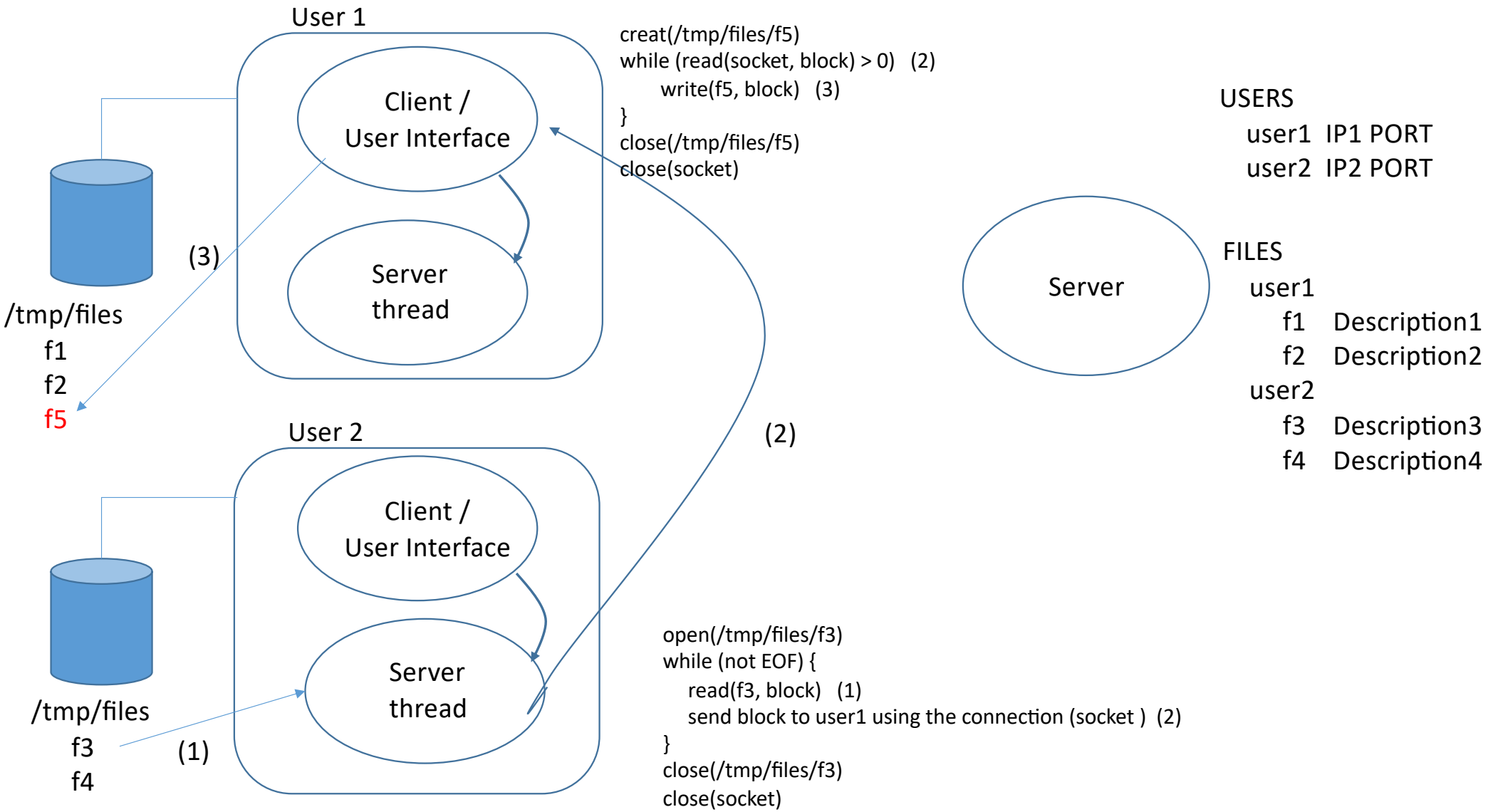


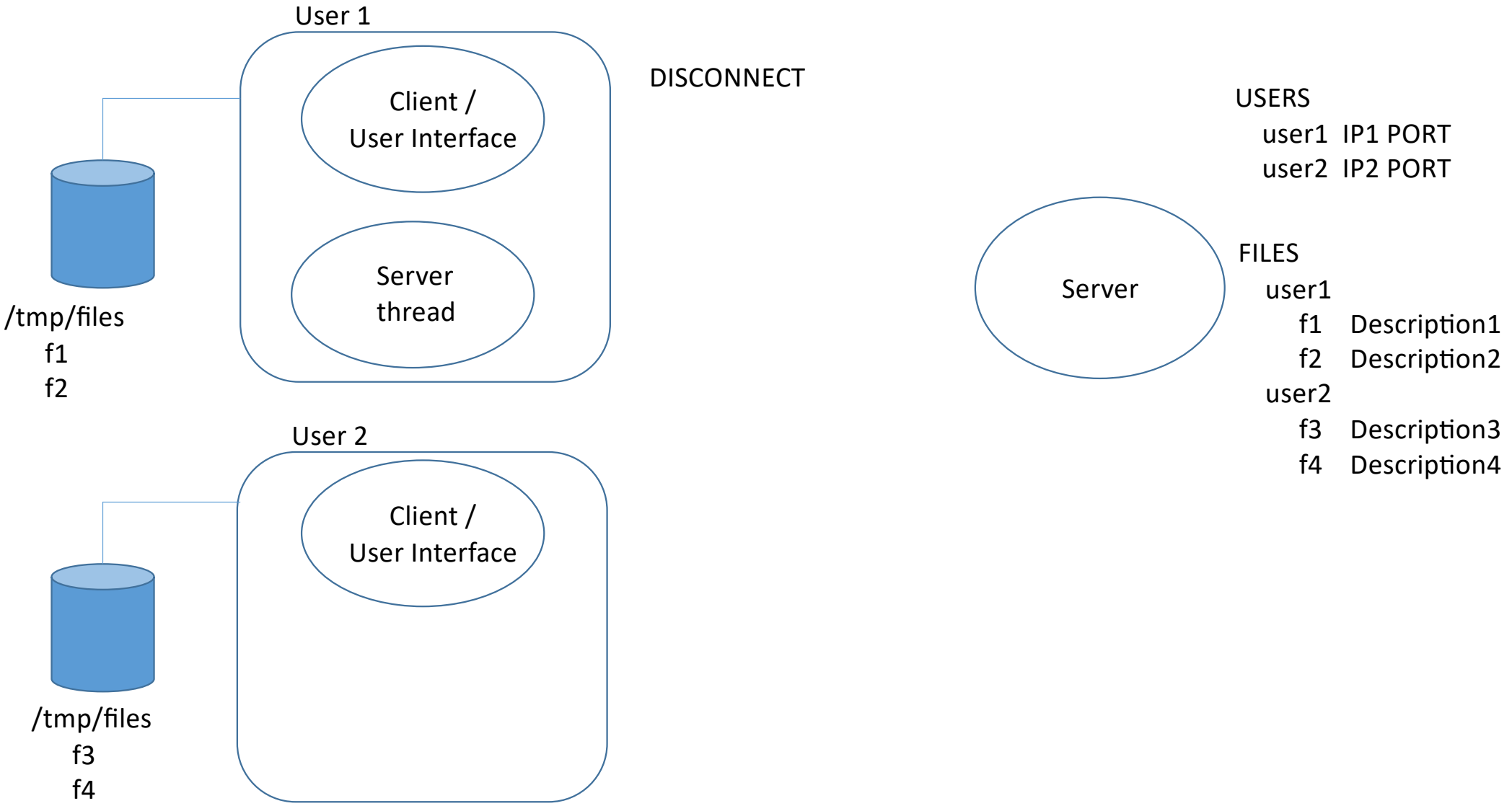
USERS
user1 IP1 PORT
user2 IP2 PORT

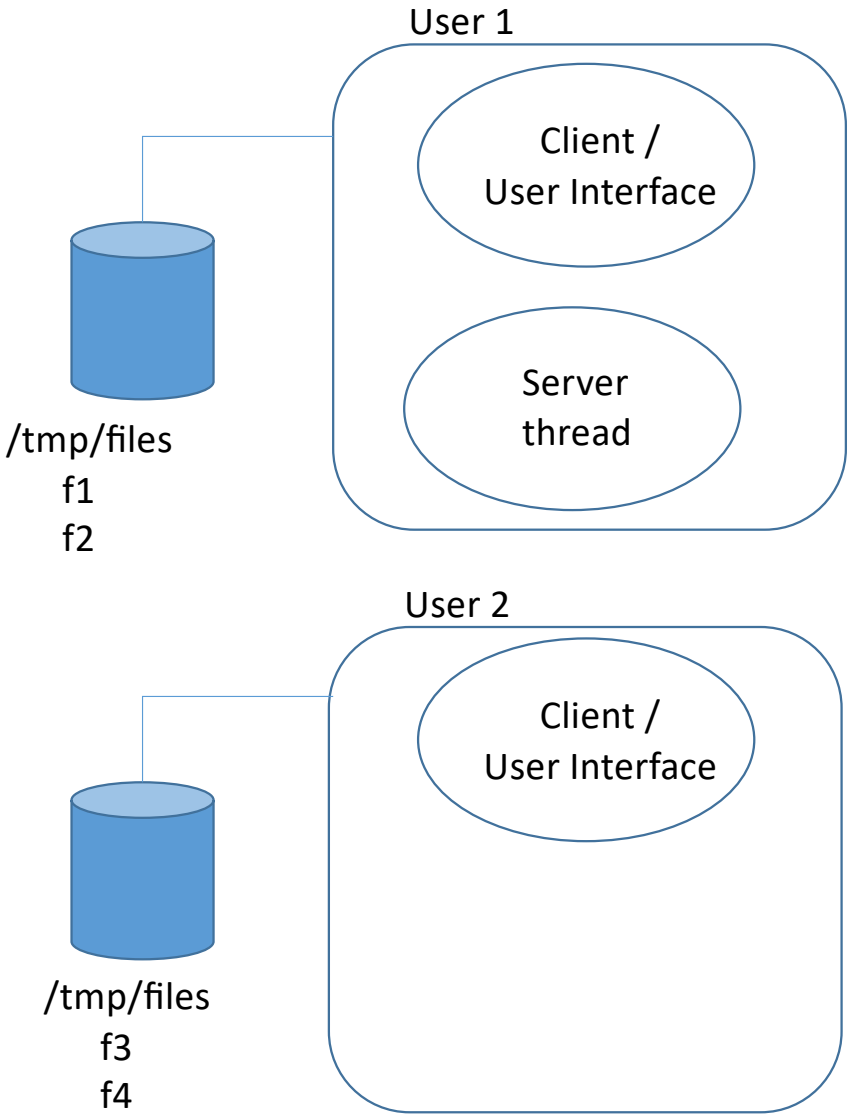
FILES
user1
f1 Description1
f2 Description2
user2
f3 Description3
f4 Description4


User 1
Client /
User Interface
Server
thread
/tmp/files
f1
f2
f5
User 2
Client /
User Interface
Server
thread
/tmp/files
f3
f4
(1)
(2)
Server
USERS
user1 IP1 PORT
user2 IP2 PORT
FILES
user1
f1 Description1
f2 Description2
user2
f3 Description3
f4 Description4
open(/tmp/files/f3)
while (not EOF) {
read(f3, block) (1)
send block to user1 using the connection (socket) (2)
}


User 1
Client /
User Interface
Server
thread
/tmp/files
f1
f2
f5
(3)
creat(/tmp/files/f5)
while (read(socket, block) > 0) (2)
write(f5, block) (3)
}
User 2
Client /
User Interface
Server
thread
/tmp/files
f3
f4
(1)
(2)
open(/tmp/files/f3)
while (not EOF) {
read(f3, block) (1)
send block to user1 using the connection (socket) (2)
}
Server
USERS
user1 IP1 PORT
user2 IP2 PORT
FILES
user1
f1 Description1
f2 Description2
user2
f3 Description3
f4 Description4


User 1
Client /
User Interface
Server
thread
/tmp/files
f1
f2
f5
(3)
creat(/tmp/files/f5)
while (read(socket, block) > 0) (2)
write(f5, block) (3)
}
close(/tmp/files/f5)
close(socket)
(2)
User 2
Client /
User Interface
Server
thread
/tmp/files
f3
f4
(1)
open(/tmp/files/f3)
while (not EOF) {
read(f3, block) (1)
send block to user1 using the connection (socket) (2)
}
close(/tmp/files/f3)
close(socket)
Server
USERS
user1 IP1 PORT
user2 IP2 PORT
FILES
user1
f1 Description1
f2 Description2
user2
f3 Description3
f4 Description4


User 1
Client /
User Interface
Server
thread
/tmp/files
f1
f2
DISCONNECT
USERS
user1 IP1 PORT
user2 IP2 PORT
Server
FILES
user1
f1 Description1
f2 Description2
user2
f3 Description3
f4 Description4
User 2
Client /
User Interface
/tmp/files
f3
f4



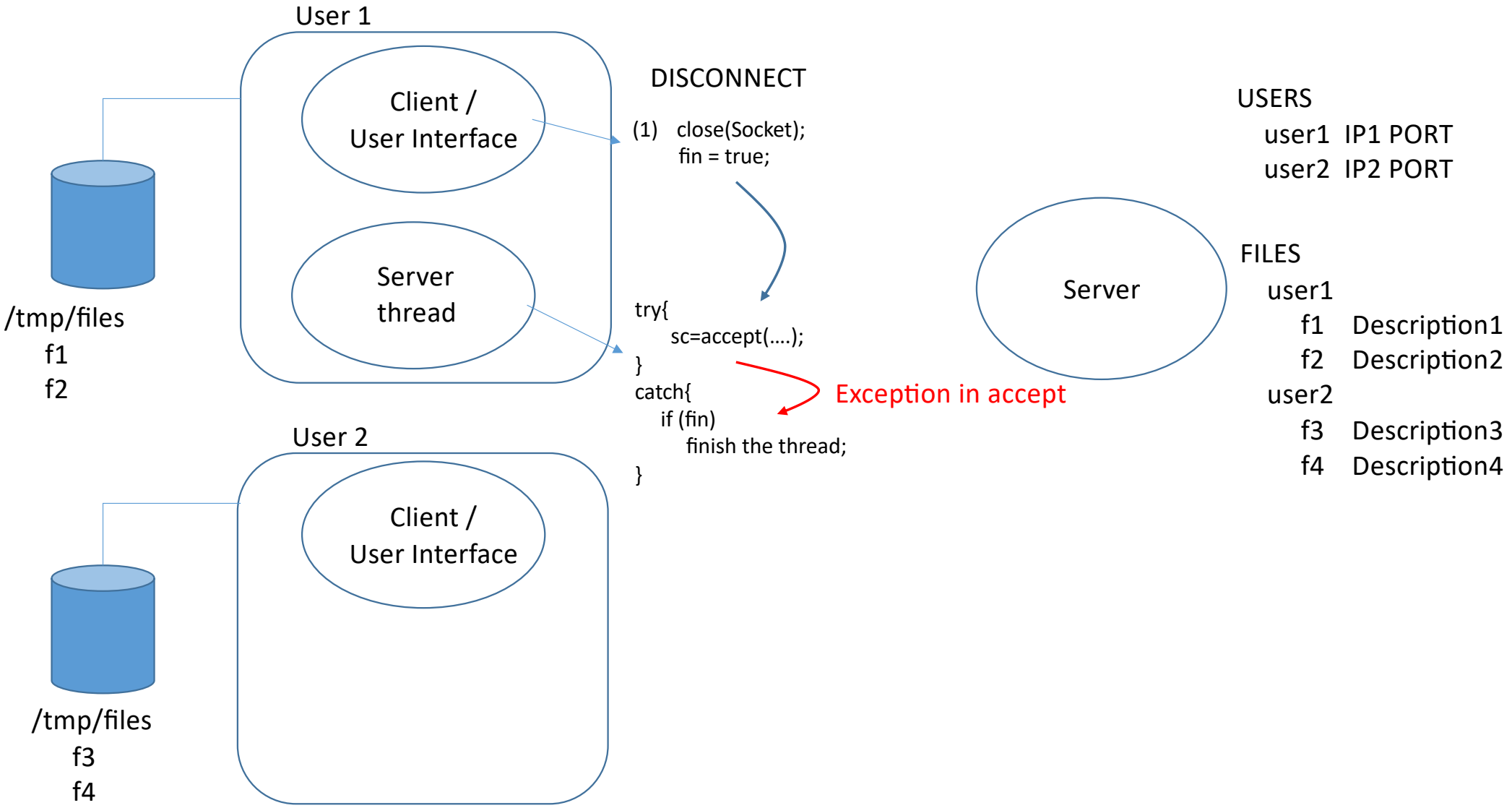
DISCONNECT

(1) close(Socket);
fin = true;

Server

USERS
user1 IP1 PORT
user2 IP2 PORT

FILES
user1
f1 Description1
f2 Description2
user2
f3 Description3
f4 Description4


User 1
Client /
User Interface
Server
thread
/tmp/files
f1
f2
User 2
Client /
User Interface
/tmp/files
f3
f4
DISCONNECT
(1) close(Socket);
fin = true;
try{
sc=accept(….);
}
catch{
if (fin)
finish the thread;
}
Exception in accept
Server
USERS
user1 IP1 PORT
user2 IP2 PORT
FILES
user1
f1 Description1
f2 Description2
user2
f3 Description3
f4 Description4

Protocolo

- Todo el protocolo se basa en enviar cadenas de caracteres:
  - "REGISTER"
  - "PORT"
  - "CONNECT“
  - Etc.
- Todas las cadenas de caracteres finalizan con el código ASCII '\0'
- Cuando se envía la cadena "REGISTER" por la red se envía:
  - "REGISTER\0"
  - Se envían 9 bytes

Envíos y recepciones de cadenas con sockets en C

- En el material de apoyo del laboratorio 3 se proporcionó la función:
  - `int readLine(int fd, void *buffer, size_t n);`
    - `fd` es el descriptor de socket (o fichero)
    - `buffer`: dirección del buffer donde se guarda la cadena
    - `n`: número máximo de bytes a leer
    - La llamada devuelve la longitud de la cadena leída.
- Ejemplo:

```
char buf[256];
readLine(sd, buf, 256);
```

Envíos y recepciones de cadenas con sockets en C

- Para enviar una cadena de caracteres se utiliza `sendMessage` proporcionada en el material de apoyo de la práctica:
- Ejemplo: para enviar REGISTER

```
char buf[256];
strcpy(buf, "REGISTER");
sendMessage(sd, buf, strlen(buf)+1);
```

Se envía '\0'

Envíos y recepciones de cadenas con sockets en Python

- Se describe en el material del tema 4

Envíos y recepciones de cadenas con sockets en Python

- Para enviar una cadena en Python

```
message = "Cadena a enviar"
message = message + "\0"
connection.sendall(message.encode())
```

Envíos y recepciones de cadenas con sockets en Python

- Para recibir una cadena en Python

```
def readString(sock):
        a = ''
        while True:
                msg = sock.recv(1)
                if (msg == b'\0'):
                        break;
                a += msg.decode()
        return(a)
```