# 3. Ejercicio 2. Extensión del chat p2p con broker

El enunciado de este segundo ejercicio es muy breve, pues se basa en otro que ya has hecho en una de las sesiones de prácticas.

Se trata simplemente de **extender** el chat p2p para añadir un nuevo comando al *peer*, de modo que el usuario ahora pueda escribir algo como:

```
/BROADCAST Mensaje importante a la comunidad!
```

y ese mensaje sea entregado a todos los peers registrados en ese momento en el broker.

## 3.1. Detalles

Cuando el peer detecta que el usuario ha introducido el comando `/BROADCAST`, envía al broker un mensaje de tipo `QUERY` especial. En ese mensaje el nick por el que pregunta es la cadena `"\n"` (este es un nick que ningún usuario podría registrar fácilmente, por eso hemos elegido esa cadena).

Cuando el broker recibe este QUERY especial debe responder con la IP y puerto de todos los peers. Esto requiere un nuevo tipo de respuesta en el protocolo, pero es muy sencillo de implementar:

- El servidor envía el código `CMD_OK` para indicar que la respuesta es positiva.
- Envía seguidamente un entero (short) con el número de usuarios registrados.
- Posteriormente itera ese número de veces para enviar un endpoint cada vez.

El peer, cada vez que recibe uno de estos endpoints lo usa para enviar el mensaje a ese peer. Recuerda que el mensaje llevará como prefijo el nick del propio emisor, y en este caso se añadirá también el texto "(BROADCAST)" para que el receptor sepa que el mensaje es un mensaje de difusión y no un mensaje privado. Así por ejemplo, el mensaje podría ser "Bob (BROADCAST)> Hola a todos".

## 3.2. Ficheros a modificar

Debes modificar los siguientes ficheros (todos ellos forman parte de la práctica 5.1 y están en el Campus Virtual):

- `protocolo.c` Añade dentro de la función `CrearSocketServidorTCP()` las línea siguientes, justo tras la la línea `sock=ret`:

```
int reuse = 1;
if (setsockopt(sock, SOL_SOCKET, SO_REUSEADDR,
                (const char*)&reuse, sizeof(reuse)) < 0)
    perror("setsockopt(SO_REUSEADDR) failed");

#ifdef SO_REUSEPORT
if (setsockopt(sock, SOL_SOCKET, SO_REUSEPORT,
                (const char*)&reuse, sizeof(reuse)) < 0)
    perror("setsockopt(SO_REUSEPORT) failed");
#endif
```

  Esto es necesario para que el broker pueda reiniciarse sin esperar a que el socket se cierre, facilitando el poder volver a lanzar el servidor cuando este muera por cualquier motivo.

- `servidor.c`. En la práctica 5.2 este fichero se llamaba `servidor-chat.c`. Renombralo como `servidor.c` y realiza las modificaciones necesarias para incluir la lógica del nuevo comando QUERY con nick especial, que retorna todos endpoints registrados.

- `cliente-con-servidor.c`. Este código no estaba en la práctica 5.1, pues era el que tú debías escribir a partir del `cliente-chat-simple.c`. Si no habías completado esa práctica, deberás hacerlo ahora. Y sobre esa versión completada añades la nueva lógica para tratar con el comando `/BROADCAST`.

Los ejecutables resultantes se llamarán `cliente-con-servidor` (este ese el *peer*) y `servidor` (este es el *broker*).

## 3.3. Pruebas

- En una terminal lanzas al servidor (broker), el cual quedará escuchando en el puerto predefinido 11111. En esa terminal irán saliendo mensajes de depuración, que ya estaban en el código de `servidor.c` (no los elimines).
- En otras terminales lanzas varios peers. Prueba que el comando `/CHAT nick` sigue funcionando correctamente y que el texto que escribas después es recibido por el peer con ese nick.
- En cualquiera de los peers (digamos en uno llamado "Bob") escribe `/BROADCAST Hola a todos`.
- En el servidor saldrá un mensaje de depuración "Solicitados datos del nick |" y en otra línea otro `|` (pues entre las barras aparece el nick solicitado y en este caso es un retorno de carro).
- En todos los peers conectados saldrá el mensaje "Bob (BROADCAST)> Hola a todos".

- Prueba a arrancar un peer nuevo, y escribe de nuevo un broadcast desde Bob. Comprueba que el mensaje llega también al nuevo.
- Prueba a lanzar un peer con un nombre que ya estuviera registrado para comprobar que el broker lo rechaza. El peer debe finalizar si la respuesta a `JOIN_CMD` es `CMD_ERROR`.
- Verifica que cuando un peer se desconecta, se desregistra correctamente ante el broker. Si no esto no fuera así, lo detectarías porque al intentar lanzar de nuevo el peer con el mismo nick no le dejaría por estar todavía registrado. Si puedes lanzarlo sin problemas, es que el desregistro del anterior estaba bien.