Sistemas Distribuidos

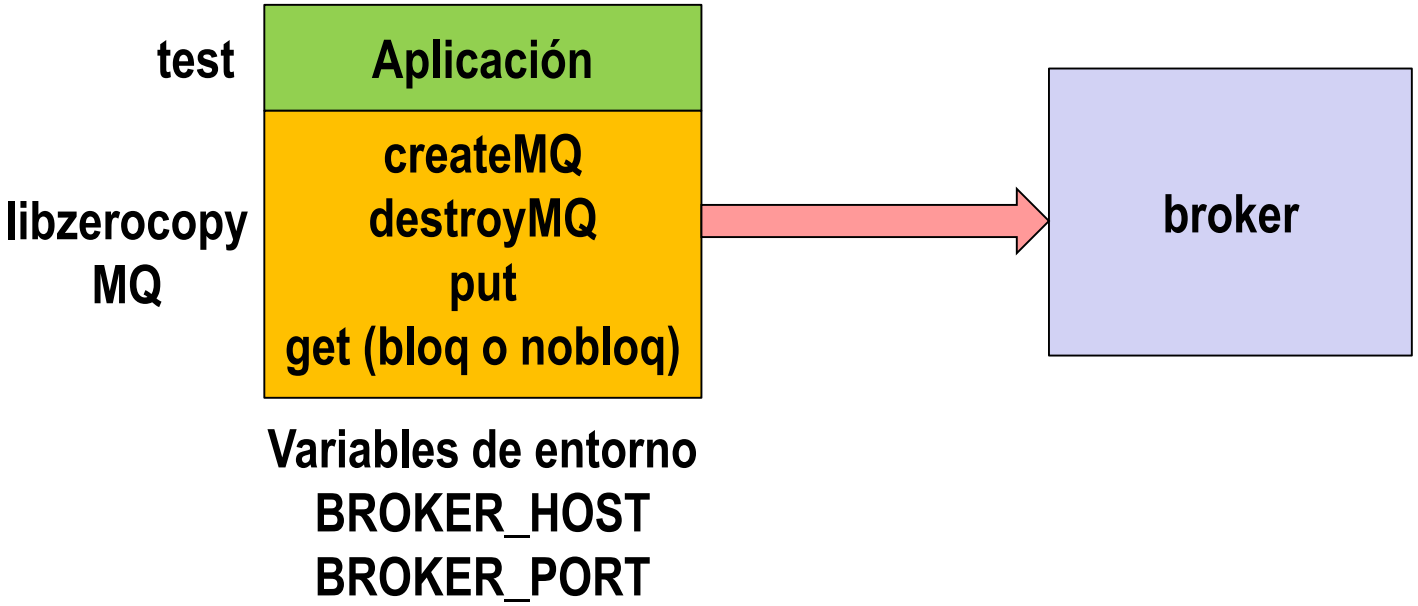
1ª práctica individual

ZerocopyMQ

Objetivo

- Sistema de colas de mensajes básico con *zerocopy*
- Requisitos **obligatorios** (debe funcionar en local y remoto):
  - Uso de *broker*
  - Uso obligatorio de tipos de datos proporcionados: diccionario y cola
  - Mensajes de aplicaciones de cualquier tipo (pueden ser binarios)
  - No límite en nº colas y de mensajes por cola
  - Tamaño máx. mensaje de app 2^{32}; longitud máx. nombre cola 2^{16}
    - Recuerde que en C cadena de caracteres debe terminar con nulo
  - Solución *zerocopy*: no copias de nombre de cola ni de mensajes
    - No múltiples envíos porque producen fragmentación
  - Optimizar ancho de banda gastado
  - Entorno heterogéneo
- Grados de libertad en el diseño:
  - Formato de mensajes, uso de conexiones persistentes
  - Servicio secuencial, concurrente o dirigido por eventos

Arquitectura de zerocopyMQ



API ofrecida a apps; Ejecución pruebas

```
int createMQ(const char *cola);
int destroyMQ(const char *cola);
int put(const char *cola, const void *mensaje, uint32_t tam);
int get(const char *cola, void **mensaje, uint32_t *tam, bool blocking);
```

---

triqui3: cd broker; make
triqui3: ./broker 12345

---

triqui4: cd test; make
triqui4: export BROKER_PORT=12345; export BROKER_HOST=triqui3
triqui4: ./test

---

triqui2: cd test; make
triqui2: export BROKER_PORT=12345; export BROKER_HOST=triqui3
triqui2: ./test

Fases

- 1ª fase (6 puntos): no GET bloqueante ni caídas de nodos
  - Punto de partida: usar ejemplos de sockets de página web asignatura
    - Código de servidor para *broker.c*
    - Código de cliente para *libzerocopyMQ.c*

    http://laurel.datsi.fi.upm.es/~ssoo/sockets/
  - Revise ejemplos de *zerocopy* en la página web de asignatura

    http://laurel.datsi.fi.upm.es/~ssoo/SD.dir/zerocopy.tgz
  - Cuidado con el problema de las recepciones incompletas

    http://laurel.datsi.fi.upm.es/~ssoo/sockets/7_recepcion_completa/
- 2ª fase (3 puntos): GET bloqueante pero sin caídas de nodos
  - Cliente se bloquea hasta que le llegue mensaje
  - Si se destruye cola, deben devolver error los clientes bloqueados
- 3ª fase (1 punto): tratamiento de caídas de nodos
  - Si cliente bloqueado caído, mensaje debe entregarse a otro
  - Y almacenarse en el *broker* si todos los clientes bloqueados caídos