Seminario 5: ZMQ

Uno de los problemas a la hora de crear grandes sistemas distribuidos es el descubrimiento. Este problema consiste en cómo cada uno de los componentes distribuidos puede conocer a las demás piezas del sistema.

Para este problema hay distintas soluciones, desde soluciones más sencillas donde el descubrimiento está escrito a fuego en el código de cada componente hasta soluciones donde el descubrimiento se hace dinámicamente y permite dotar al sistema distribuido de una elasticidad y escalabilidad sin necesidad de intervención administrativa.

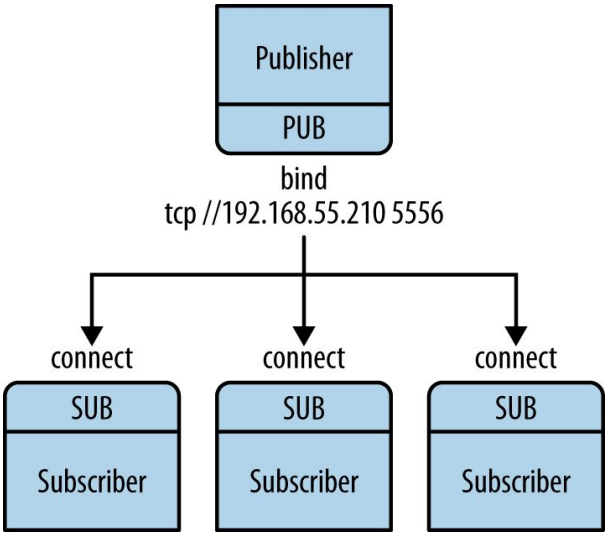

Ilustración 3.1: Diagrama único publicador, múltiples suscriptores

En la práctica, adoptar medidas fáciles o escritas en código puede derivar a arquitecturas pesadas y frágiles. Imaginemos una situación donde existe un publicador y varios suscriptores abonados a este publicador (Ilustración 3.1). Cada suscriptor tiene configurado al publicador en el código. El acceso de suscriptores es dinámico, ya que todos conocen al publicador. Si en cualquier caso se quiere añadir un nuevo suscriptor, esta tarea ya no es tan trivial.

Una primera solución al problema, y la más sencilla, es la utilización de un elemento intermediario que permita hacer la distribución hacia los publicadores. Este elemento es más comúnmente conocido como broker, aunque ZMQ no implementa este elemento en su librería es relativamente sencillo crear intermediarios de este tipo.

La adición de un elemento que actúe como actúa un proxy HTTP resolvería el problema en nuestro ejemplo. El proxy colocado abriría un puerto suscriptor para recibir información de los publicadores y abriría un puerto publicador para publicar hacia los suscriptores (Ilustración 3.2).

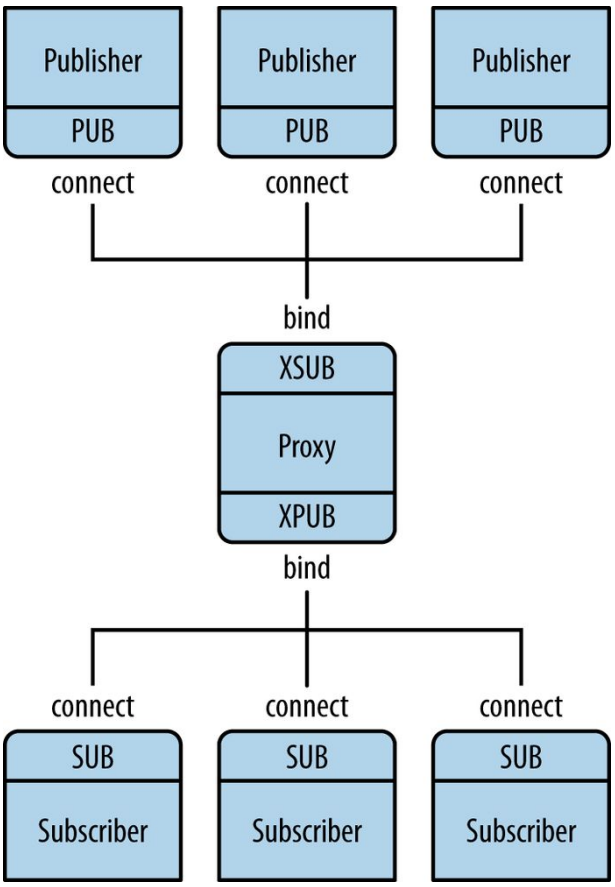

Ilustración 3.2: Diagrama publicadores y suscriptores con proxy

La implementación del ejemplo del xpub/xsub se puede encontrar en el repositorio:

https://github.com/carvido/sad-sem5.git

Se ha implementado un descubrimiento dinámico, con un heartbeat de 5 segundos donde el componente pub anuncia su nombre (en este caso el puerto para el socket rep) que permite a los componentes sub hacer peticiones req a los pub.

Para la ejecución se puede utilizar el siguiente código:

```
node proxy.js
node sub.js
node sub.js
node sub.js
node pub.js 6000 '{"0":"abc","1":"def","2":"ghi","3":"jkm","4":"nlo","5":"pqr","6":"stu"}'
node pub.js 6001 '{"0":"0a","1":"1b","2":"2c","3":"3d","4":"4e","5":"5f","6":"6g"}'
```

En todos los componentes se ha modificado la señal SIGINT para cerrar el programa con un solo ctrl+c de forma ordenada (cerrando los sockets) y con un segundo ctrl+c se provocaría la finalización del proceso.