URJC Escuela de Ingeniería de Fuenlabrada

GSyC RoboticsLabURJC Programming Robot Intelligence

Sistemas Distribuidos y Concurrentes

Publicador/Subscriptor

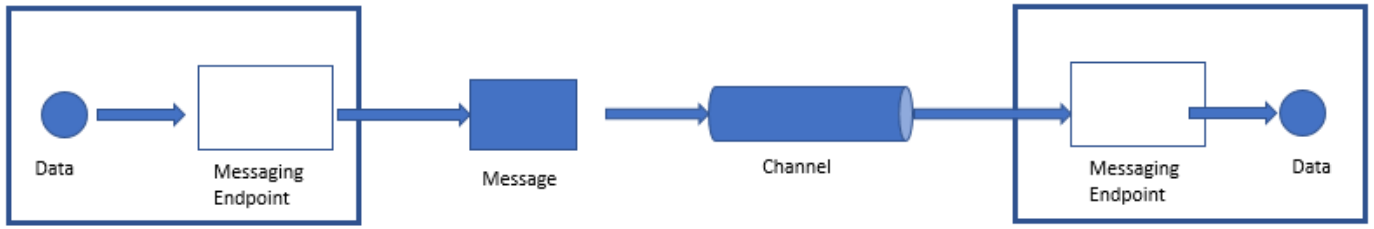
Grado en Ingeniería de Robótica Software

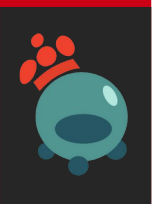
Teoría de la Señal y las Comunicaciones y Sistemas Telemáticos y Computación

Roberto Calvo Palomino
roberto.calvo@urjc.es

Introducción

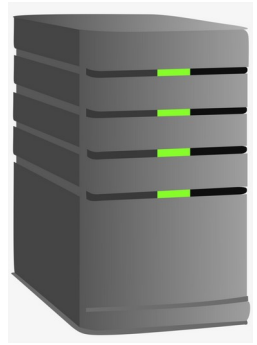
- Existen diferentes patrones para comunicar procesos.
- Pueden ser comunicaciones locales (pipes, ficheros, memoria compartida) o comunicaciones globales (sockets).
- Dependiendo de los agentes envueltos en la comunicación podemos utilizar conexiones
  - Uno a uno
  - Uno a muchos
  - Muchos a uno
  - Muchos a muchos





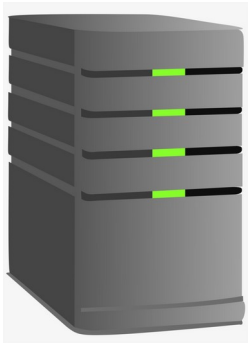
Introducción

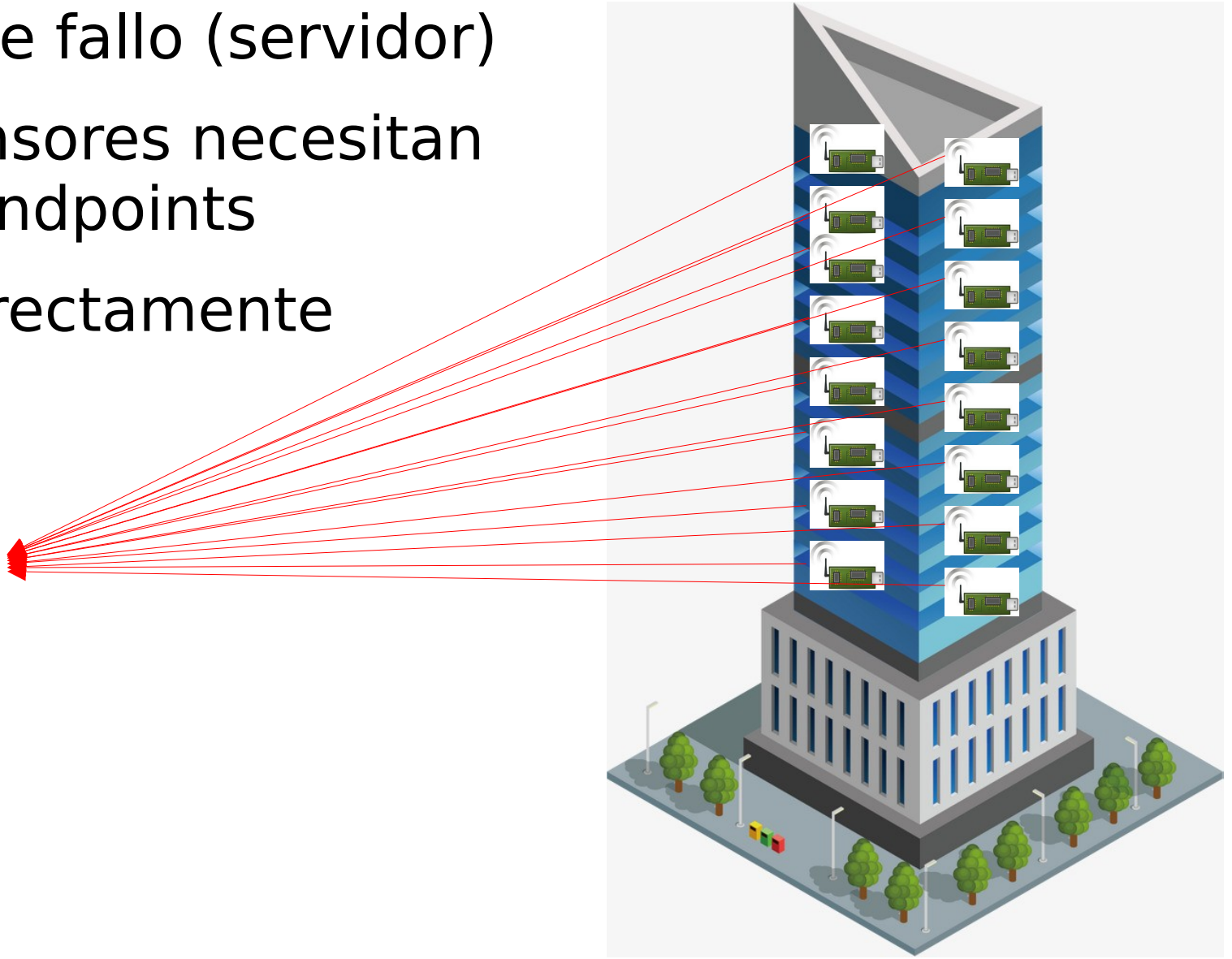
- Sistema masivo de recolección de información de sensores.
- Sensorizar temperatura de un edificio grande.



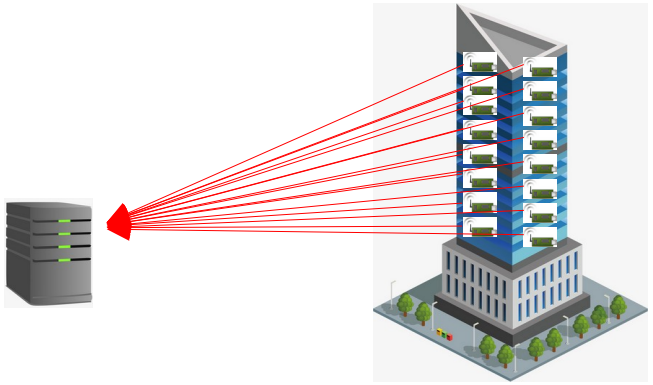
Introducción

- Único punto de fallo (servidor)
- Servidor y sensores necesitan conocer sus endpoints
- No escala correctamente
- No es flexible

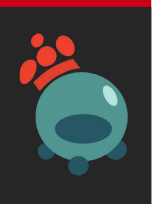




Introducción

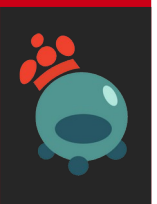


- Características que serían interesantes en este sistema:
  - Mínimo ancho de banda usado
  - Real-time (baja latencia)
  - “Push” en lugar de “Pull”
  - Fiable, incluso en redes no fiables (UDP)
  - Desacoplar software y hardware

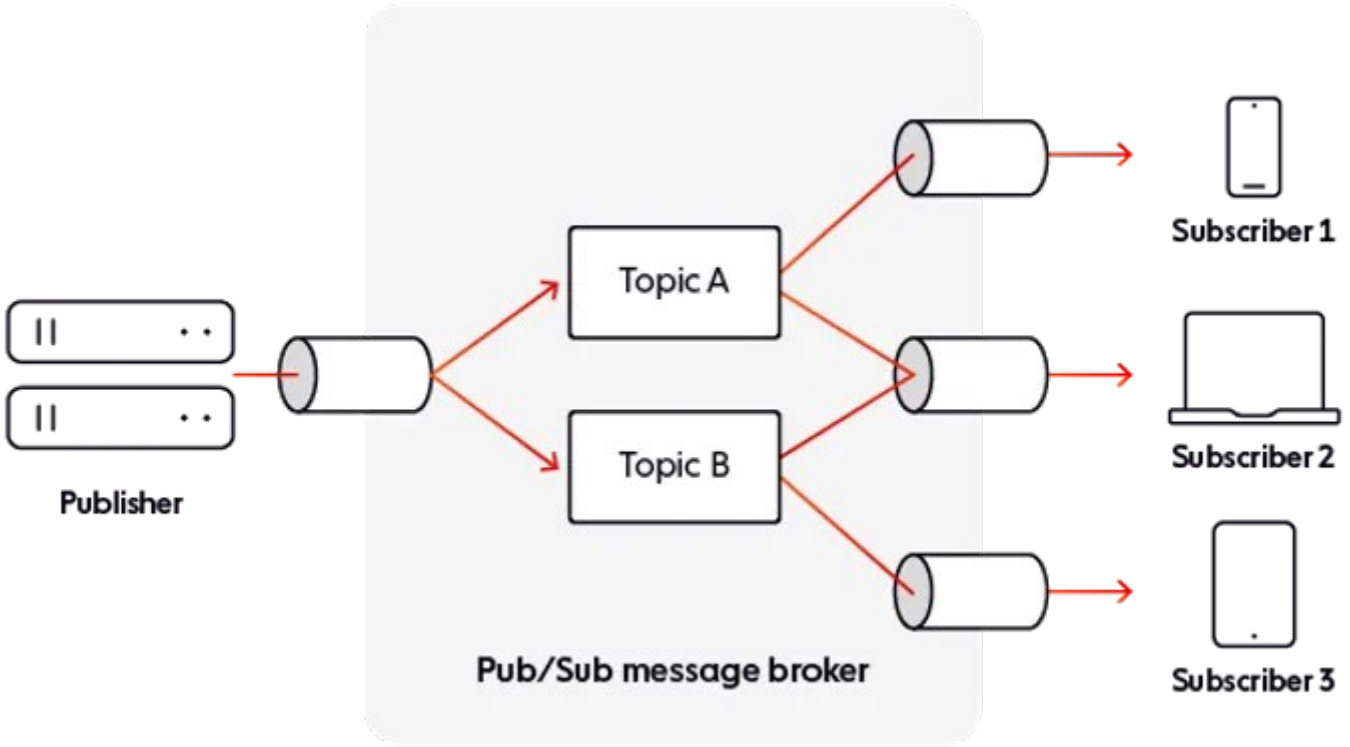


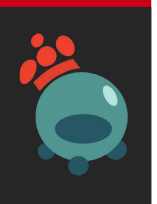
Publicador/Subscriptor

- **Publicador/Subscriptor** es un patrón de diseño muy utilizado para distribuir mensajes generados por un publicador a todos los subscriptores.
- El publicador genera mensajes con información y eventos asociados a un **topic** que envía al broker.
- El subscriptor se conecta al broker y solicita recepción de mensajes de un topic específico.
- No hay conexión directa entre publicadores y subscriptores, ni si quieran saben el número exacto de nodos en el sistema.
- Los topics permiten filtrar mensajes.
- Comunicación asíncrona.



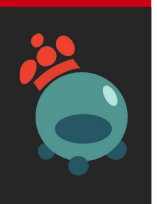
Publicador/Subscriptor





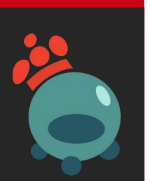
Publicador/Subscriptor

- Ventajas
  - Bajo acoplamiento entre nodos del sistema.
  - Permite una gran escalabilidad en el sistema.
  - Habilita las topologías dinámicas en la red.
  - Flexibilidad y diseño limpio.
- Desventajas
  - Las versiones centralizadas tienen un punto único de fallo
  - Cuello de botella en el broker.
  - Errores en el broker de no entrega de mensajes, no se notifican a los publicadores/subscriptores



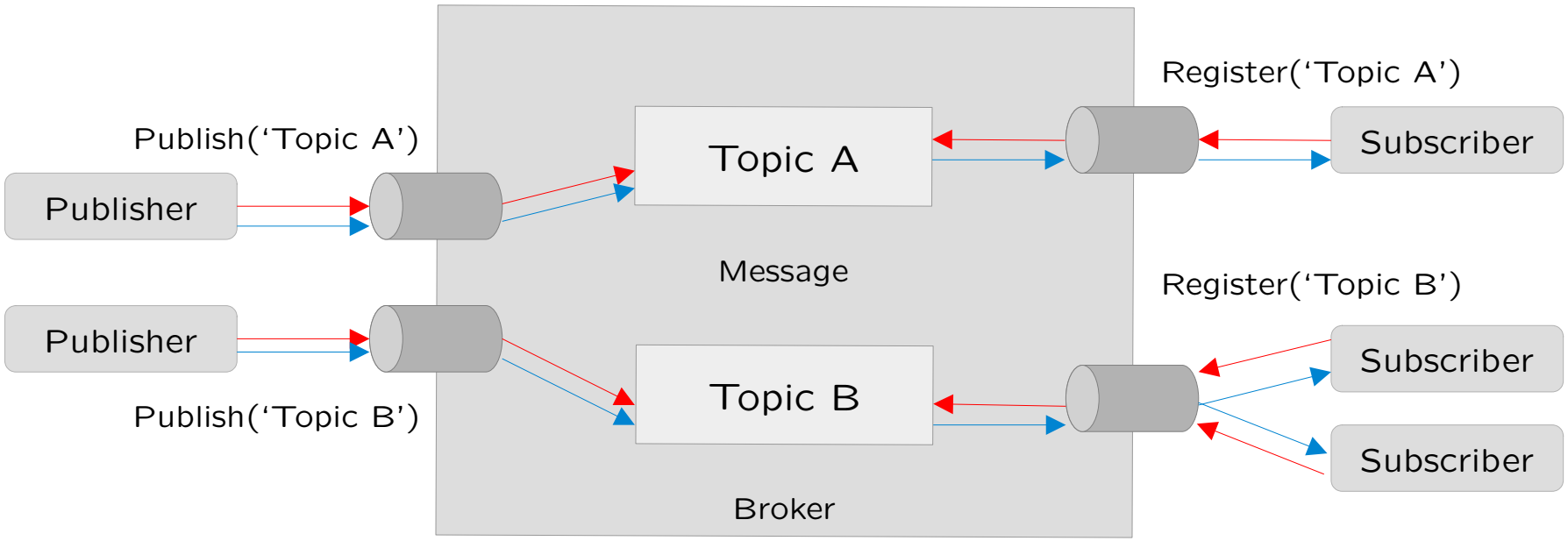
Usos

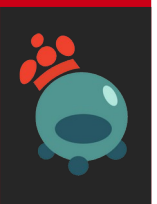
- Aplicaciones de Chat
- Notificación de eventos
- Cuando existen varias fuentes de datos (origen) y no es importante saber de donde vienen los datos.



Pub/Sub Centralizado

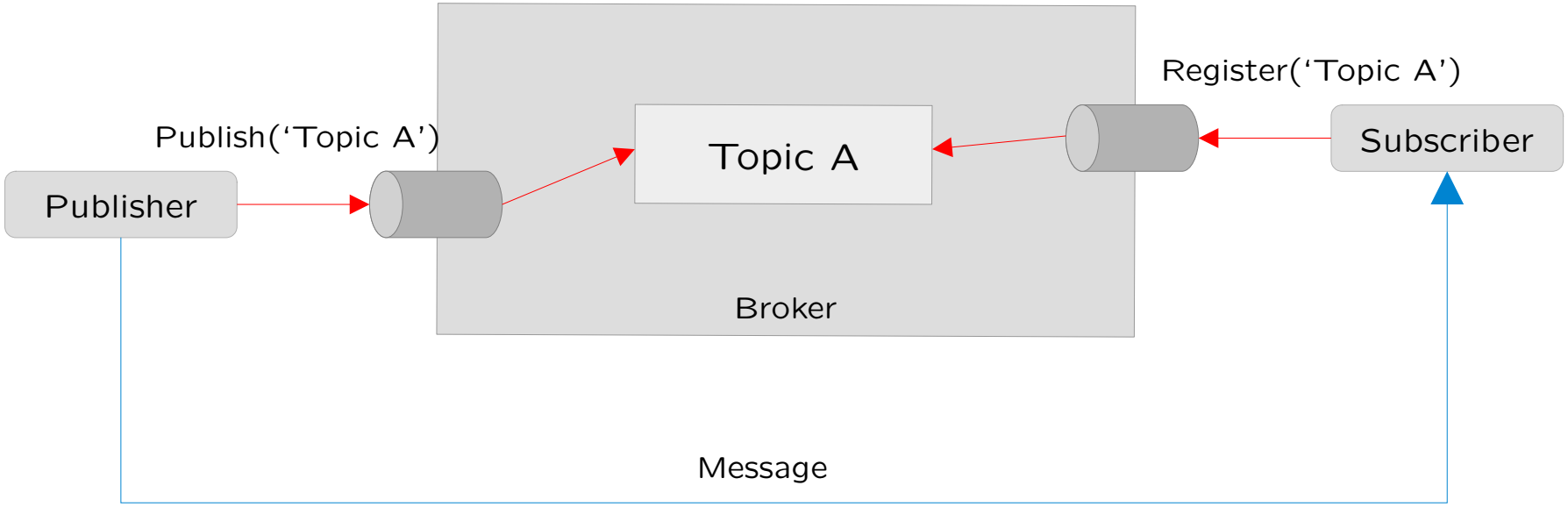
- Todo el flujo pasa por el único broker del sistema.
- Registros y subcripciones (rojo) y datos (azul)

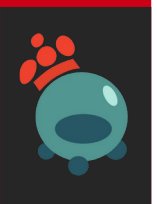




Pub/Sub Centralizado (parcialmente)

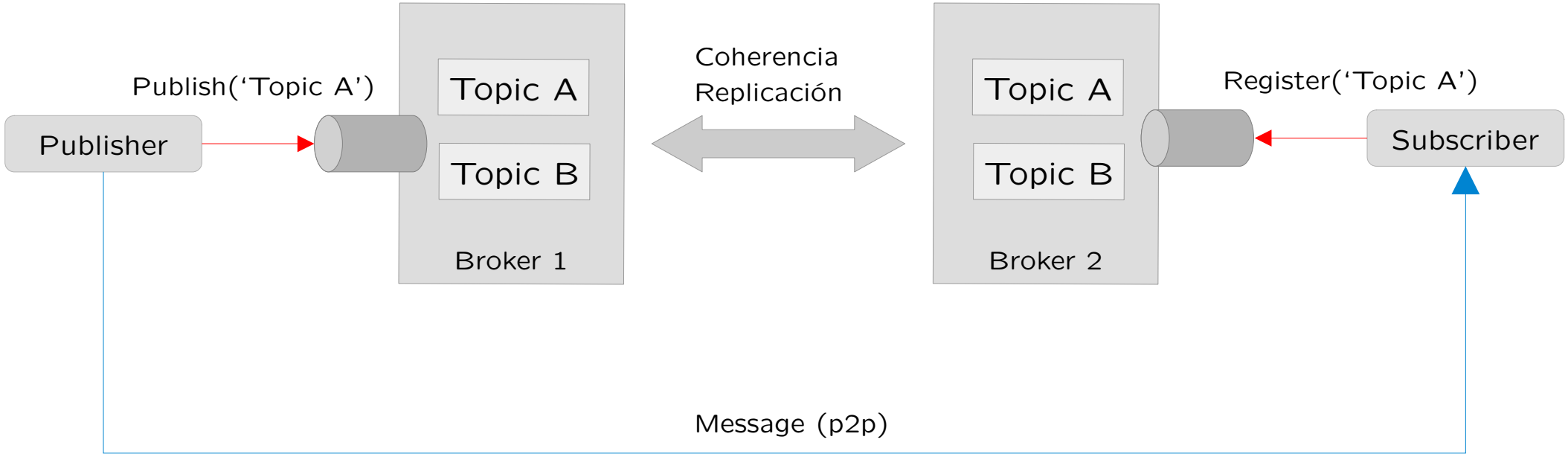
- Solo el registro y petición del topic pasa por el único broker del sistema.
- Los mensajes son directamente intercambiados entre el publicador y el subscriptor (p2p)





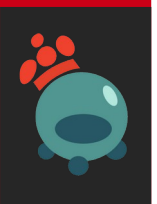
Pub/Sub Distribuido

- El servicio de registro y descrubrimiento de servicios está distribuido.
- Los mensajes son directamente intercambiados entre el publicador y el subscriptor (p2p)



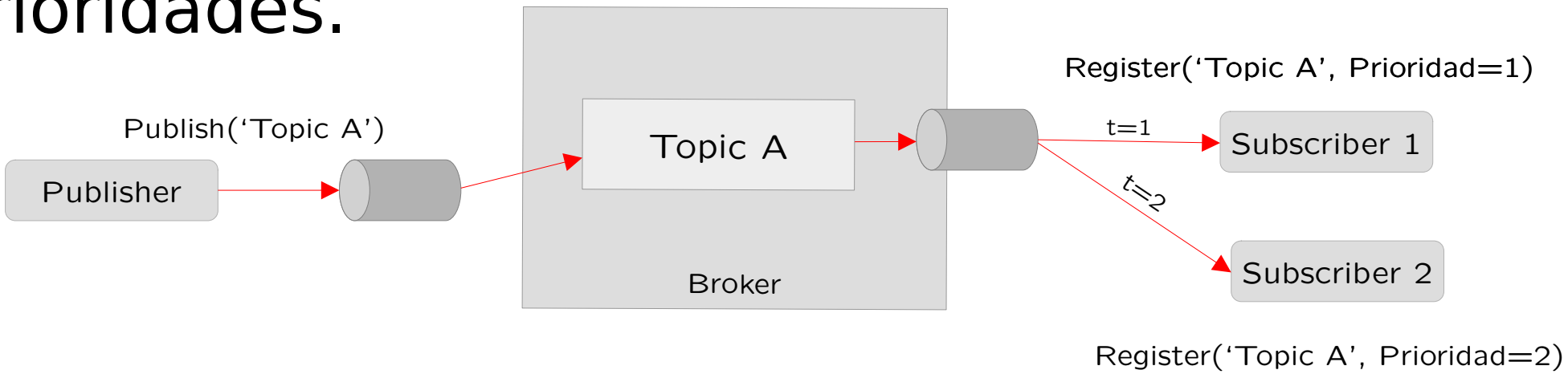
Quality of Service (QoS)

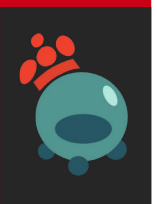
- La calidad del servicio (QoS) en aplicaciones distribuidas se refiere al rendimiento de la red/sistema desde el punto de vista del usuario final.
- Congestión de red, fallos en comunicaciones, desconexiones aleatorias pueden provocar que el mensaje no se entregue correctamente.
- ¿Qué ocurre si el mensaje publicado por el *publisher* no llega a todos los *subscribers*?
  - Orden de parar los motores de las N ruedas de un robot.
- Los sistemas pub/sub suele ofrecer diferentes niveles QoS para asegurar que los paquetes lleguen correctamente.



Quality of Service (QoS)

- Publicadores y Subscriptores pueden llevar asociados prioridades para que se traten de manera preferente frente al resto de nodos
- Subscriber1 recibe antes el mensaje que Subscriber2 debido a la configuración de las prioridades.

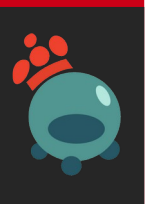




Quality of Service (QoS)

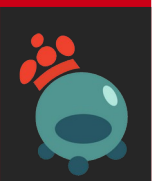
- Los subscriptores pueden registrarse en el sistema pub/sub con diferentes niveles de QoS:
  - QoS 0: Recibe el mensaje como mucho una vez.
  - QoS 1: Recibe el mensaje al menos una vez.
  - QoS 2: Recibe el mensaje exactamente una vez.

- Dependiendo del contexto de la aplicación intereserá configurar los subscriptores con diferentes niveles de QoS
- El uso de QoS implica una sobrecoste en el rendimiento del sistema.



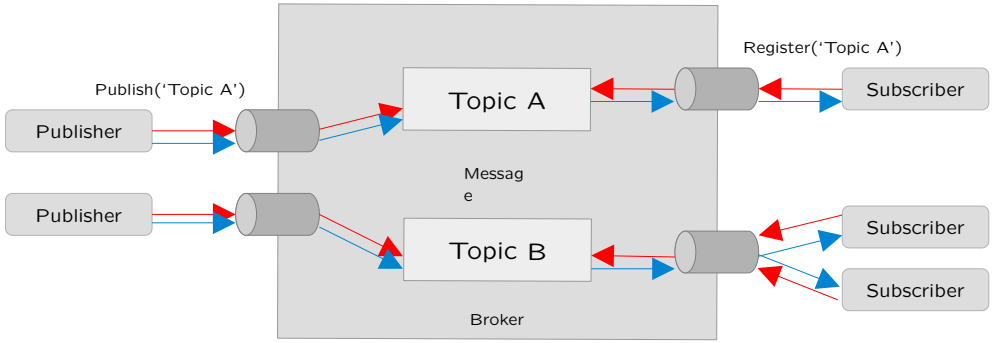
Sistemas

- Apache Kafka
  - Uber, Spotify, Shopify
- MQTT
  - Muy utilizado en IoT
- DDS
  - ROS2 y sistemas industriales
- Amazon SNS
- Corba
- Google Cloud



MQTT

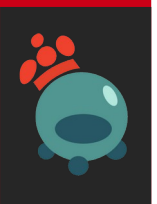
- Es un protocolo de red basado en pub/sub estandard para comunicaciones IoT
- Originalmente desarrollado por IBM (1999), ahora parte del estándar OASIS
- Se ejecuta normalmente sobre TCP/IP
- Es computacionalmente ligero.
- Concepto de broker y clientes.
- Podemos considerarlo como pub/sub centralizado



MQTT

MQTT: The Standard for IoT Messaging

MQTT is an OASIS standard messaging protocol for the Internet of Things (IoT). It is designed as an extremely lightweight publish/subscribe messaging transport that is ideal for connecting remote devices with a small code footprint and minimal network bandwidth. MQTT today is used in a wide variety of industries, such as automotive, manufacturing, telecommunications, oil and gas, etc.



Publicador/Subscriptor

Usos MQTT

Automotive

- HiveMQ: BMW Car-Sharing application relies on HiveMQ for reliable
- EMQ helps SAIC Volkswagen building IoV platform.

Consumer Products

- CASO Design creates Smart Kitchen Appliances.

Manufacturing

- MQTT Implementation on Celikler Holding's Power Plant Monitoring.

Oil & Gas

- EMQ helps IoT innovation in the petrochemical industry

Oil & Gas

- EMQ helps IoT innovation in the petrochemical industry

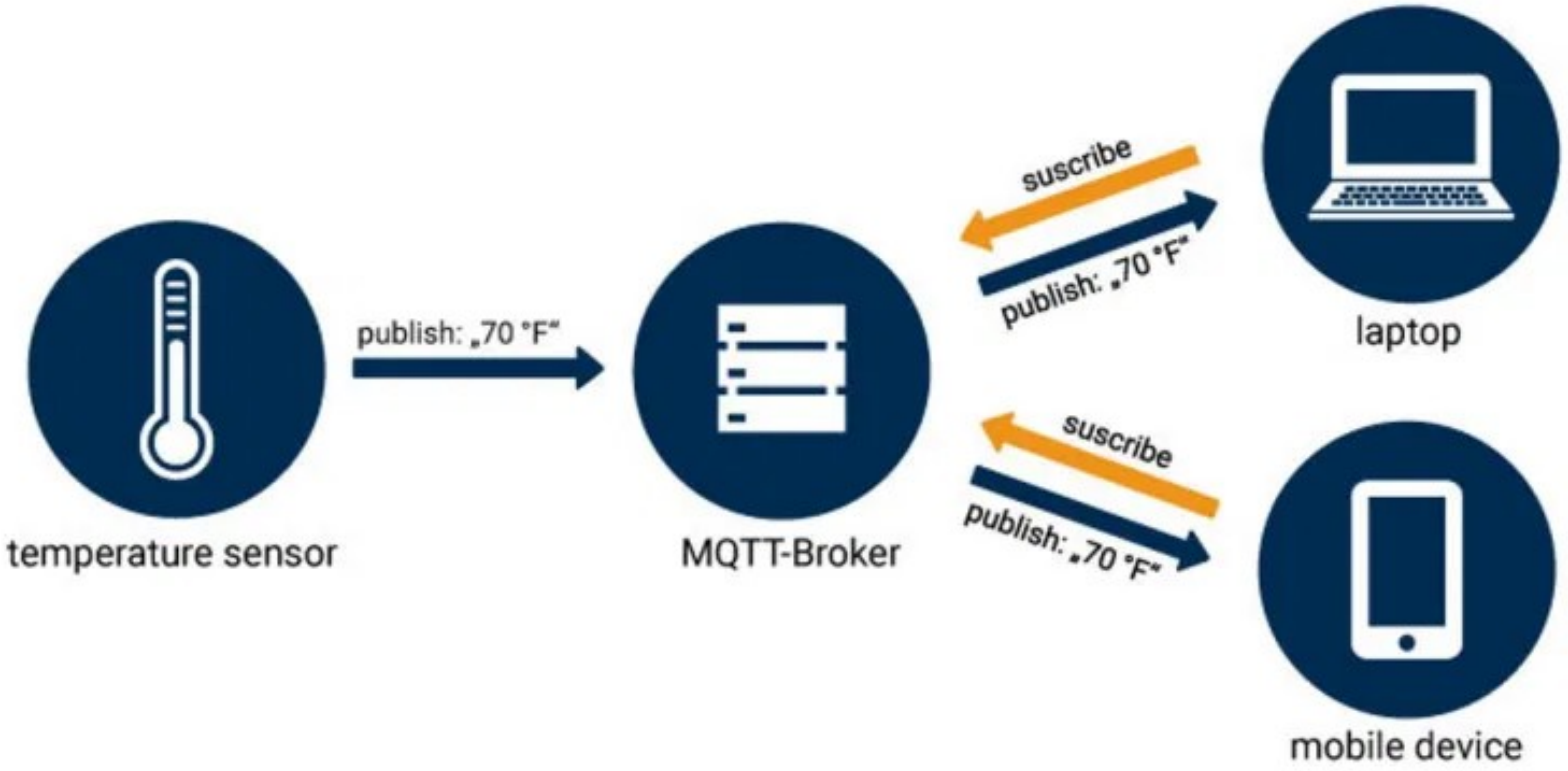
Transportation

- Deploying IoT on Germany's DB Railway System.

https://mqtt.org/use-cases/

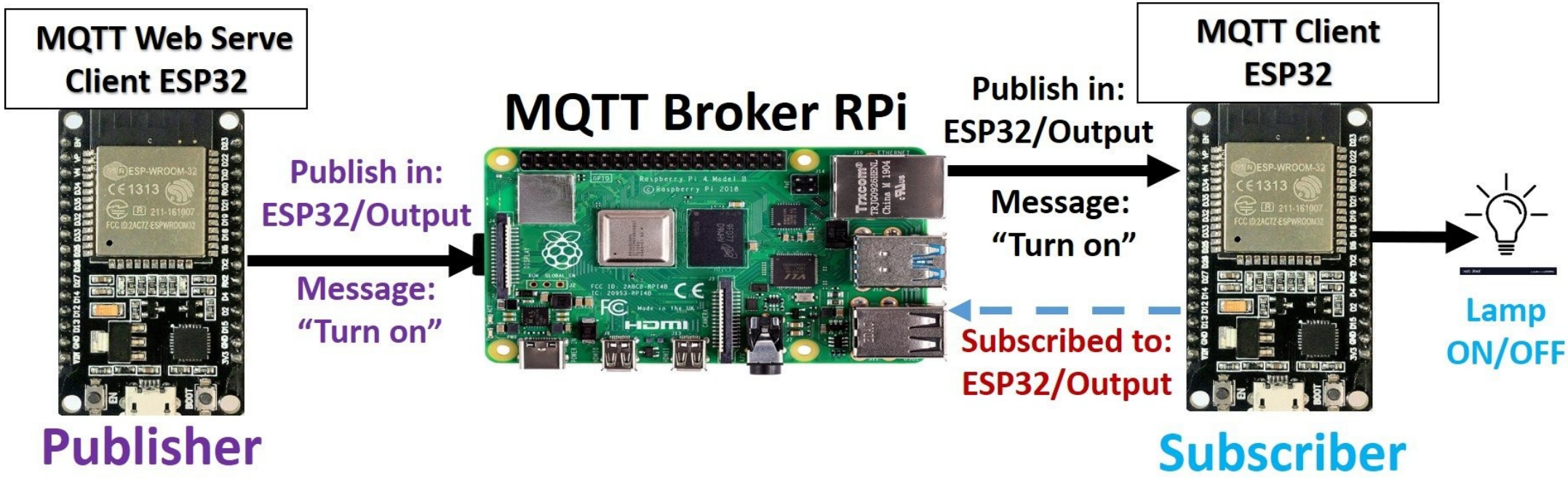
MQTT

Publicador/Subscriptor



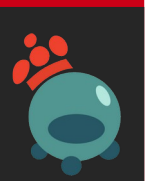
MQTT

- Ya que ofrece un protocolo muy ligero y sin mucho coste computacional por lo que es muy recomendado para IoT y micro-controladores
- ESP, Arduino, RaspberryPi.



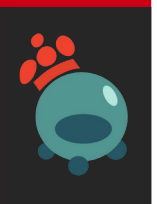
MQTT

- Ventajas
  - Eficiencia, bajo computo y consumo
  - Poco uso de red para intercambio de mensajes
  - Implementación sencilla
- Desventajas
  - Ciclos lentos de transmisiones cuando el sistema se compone por más de 200 nodos.
  - Seguridad y encriptación. Poco más que uso de TLS
  - Escalabilidad



MQTT

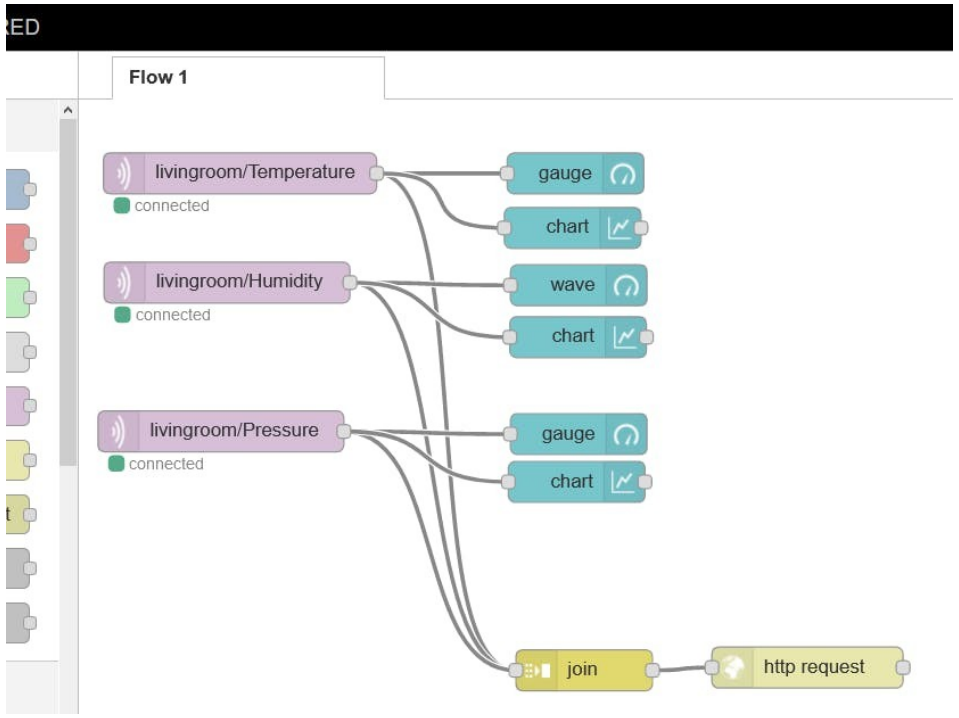
- 3 niveles de QoS (Quality of Service)
  - Disparar y olvidar (0) – Fire and Forget – At most once
  - Al menos una vez (1) – At least once
  - Exactamente una vez (2) – Exactly once



Mosquitto

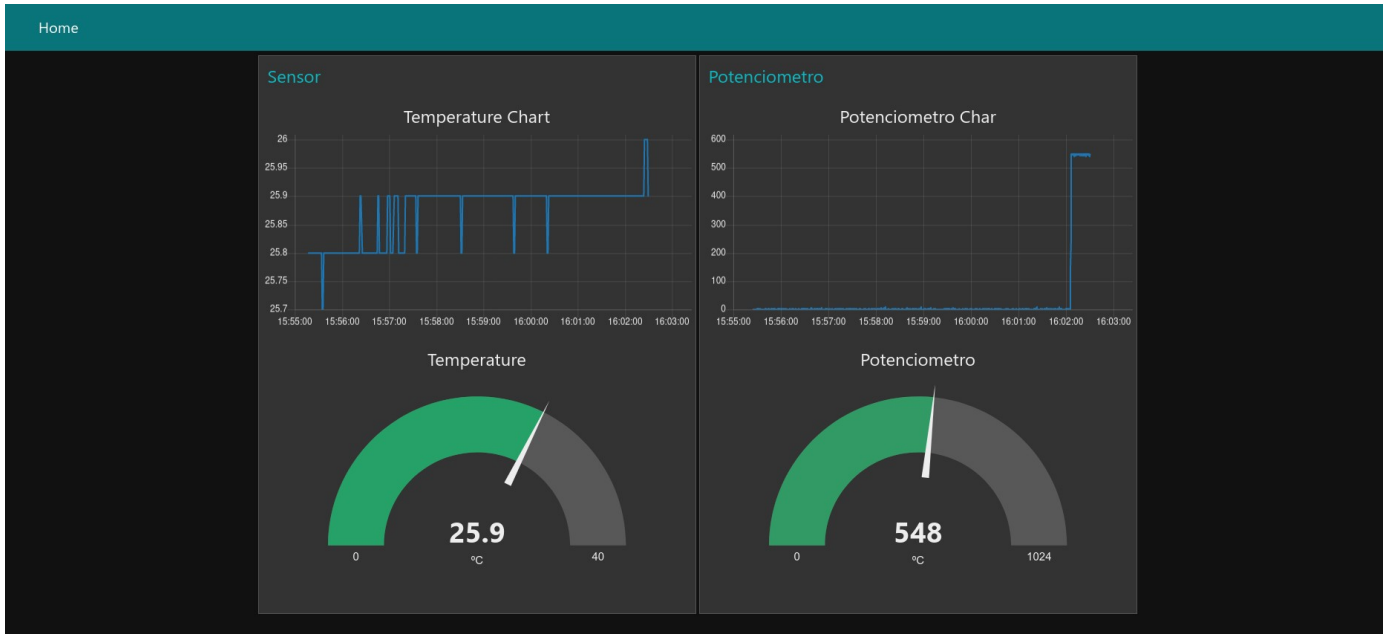
Publicador/Subscriptor

- Mosquitto es una implementación open source de MQTT, que utiliza la arquitectura pub/sub.
- Muy utilizado en sistemas linux que actua de broker.
- Posibilidad de comunicarse con nodered



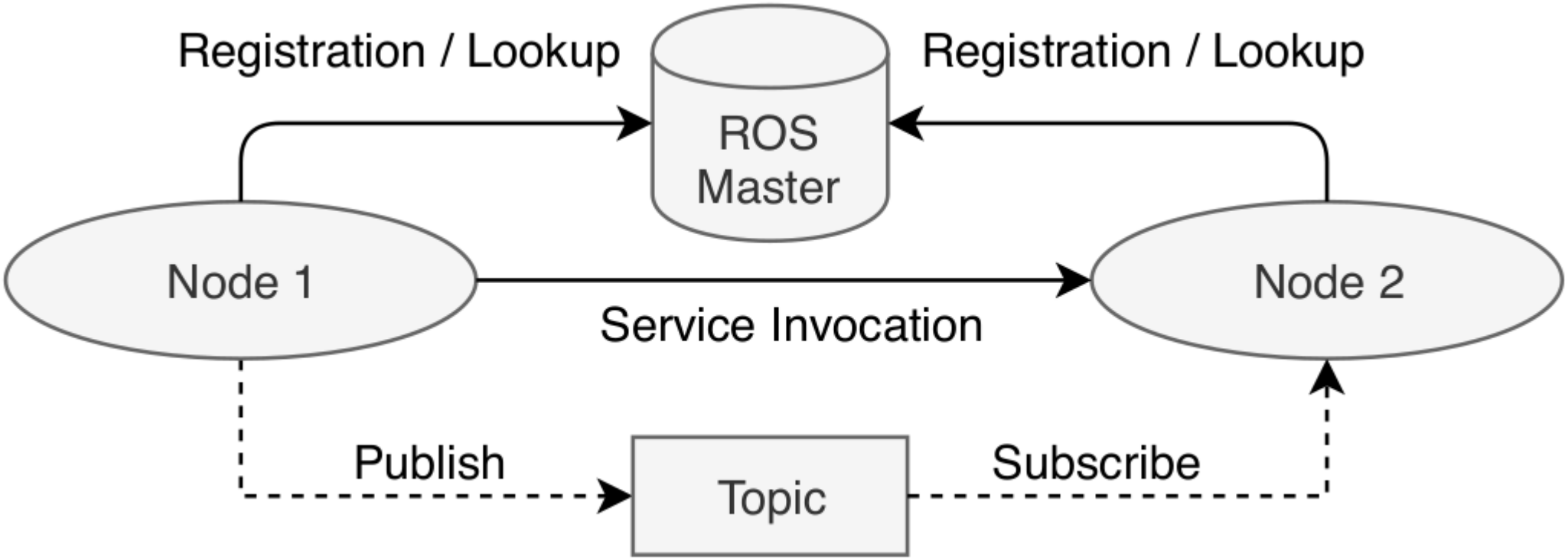
Ejemplo uso Mosquitto (MQTT)

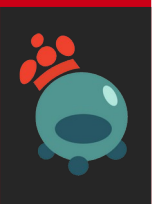
https://gitlab.etsit.urjc.es/roberto.calvo/setr/-/tree/main/MQTT-NodeRed



ROS

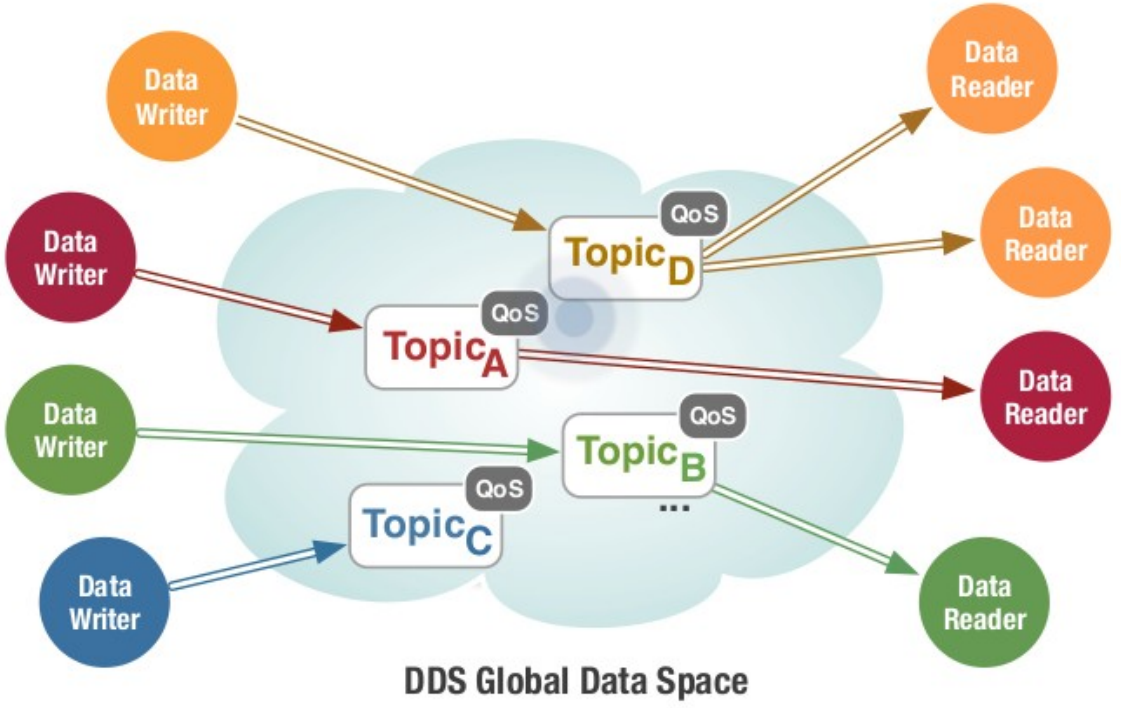
- Pub/Sub parcialmente centralizado.





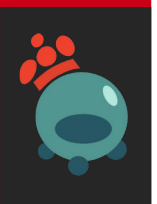
ROS2 (DDS)

- Totalmente distribuido
- Data Distributed Service
- Descubrimiento dinámic
- Real-time
- Algo rendimiento
- Seguridad avanzada
- Múltiples QoS



Bibliografía

- Publish / Subscribe Systems: Design and Principles
  - Wiley Series on Communications Networking
- The Data Distribution Service Tutorial
  - https://www.researchgate.net/publication/273136749_The_Data_Distribution_Service_Tutorial



URJC Escuela de Ingeniería de Fuenlabrada

GSyC

RoboticsLabURJC

Programming Robot Intelligence

Publicador/Subscriptor