

Laurea in Informatica

Il paradigma Publish/Subscribe

Prof. Davide Quaglia

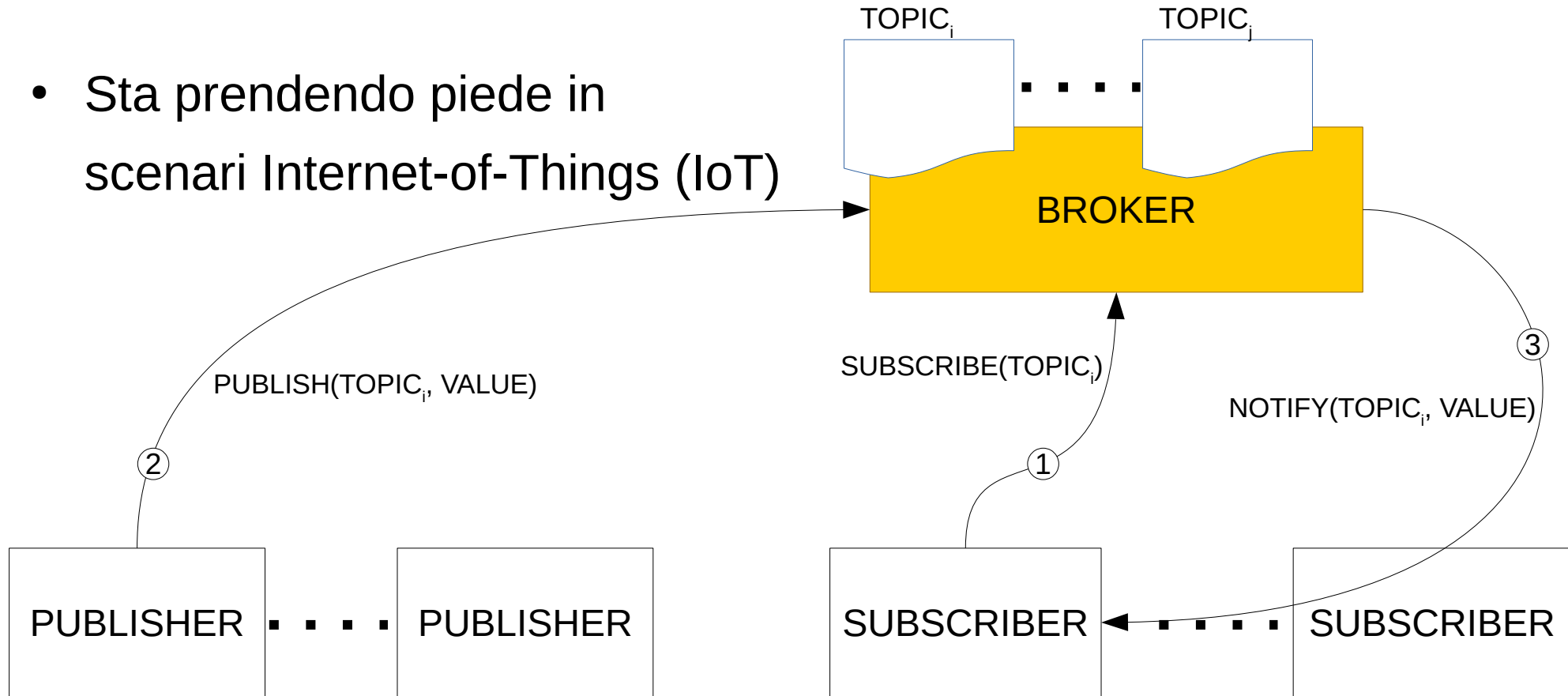


UNIVERSITÀ
di VERONA
Dipartimento
di INFORMATICA



Modello Publisher/Subscriber (Pub/Sub)

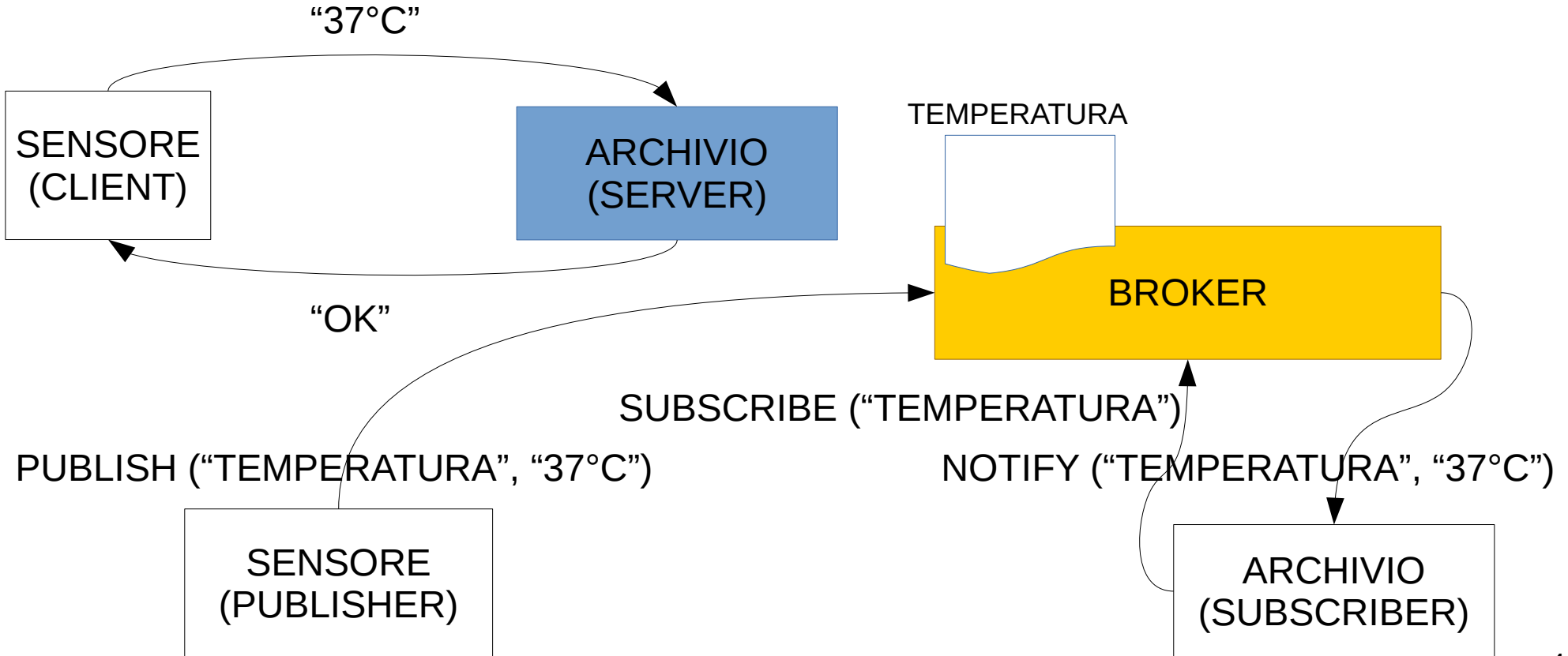
- Sta prendendo piede in scenari Internet-of-Things (IoT)



Confronto Client/Server ↔ Pub/Sub

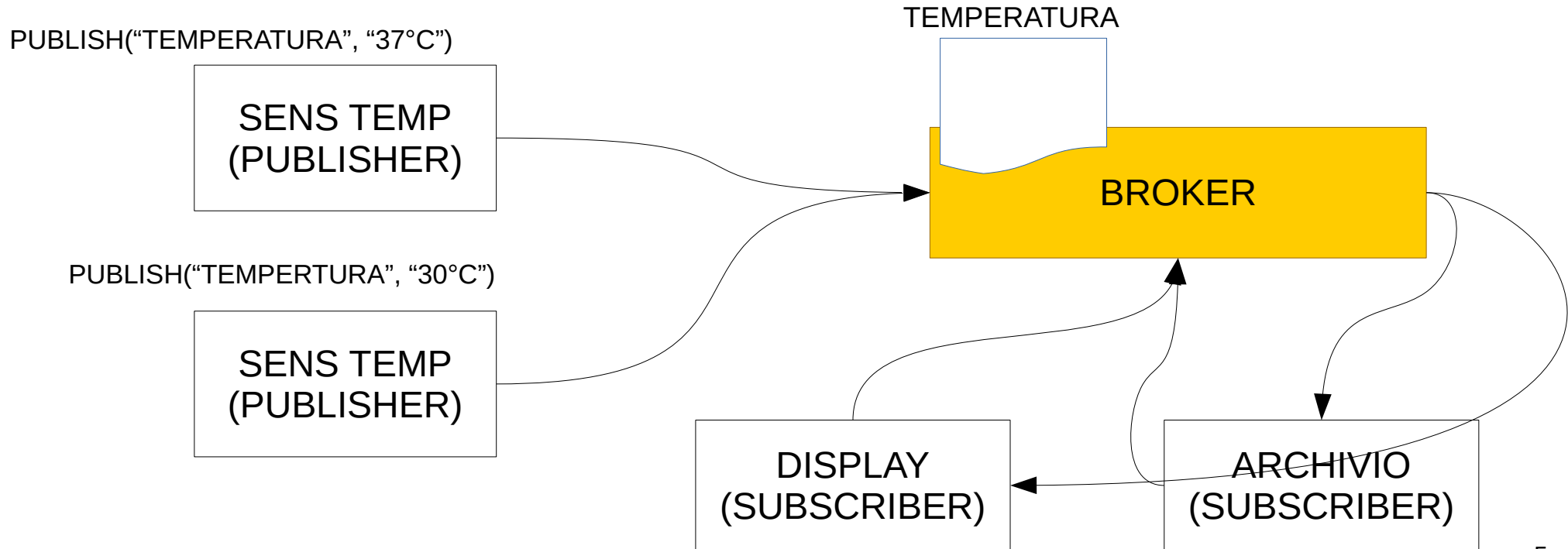
- L'host su cui gira un programma server deve avere requisiti particolari
 - Sempre acceso
 - Sempre raggiungibile su Internet
 - Quindi dotato di un indirizzo IP pubblico senza NAT e Firewall troppo stretti
 - Capace di rispondere ad un numero anche molto grande di client
- In un'applicazione pub/sub è il broker ad assumere le prerogative tipiche del server mentre i publisher e i subscriber sono come dei client
- Un'applicazione client/server si può trasformare sempre in un'applicazione pub/sub dove i client e server originari diventano più leggeri e la complessità viene scaricata sull'host che fa da broker

Esempio di trasformazione Client/Server → Pub/Sub



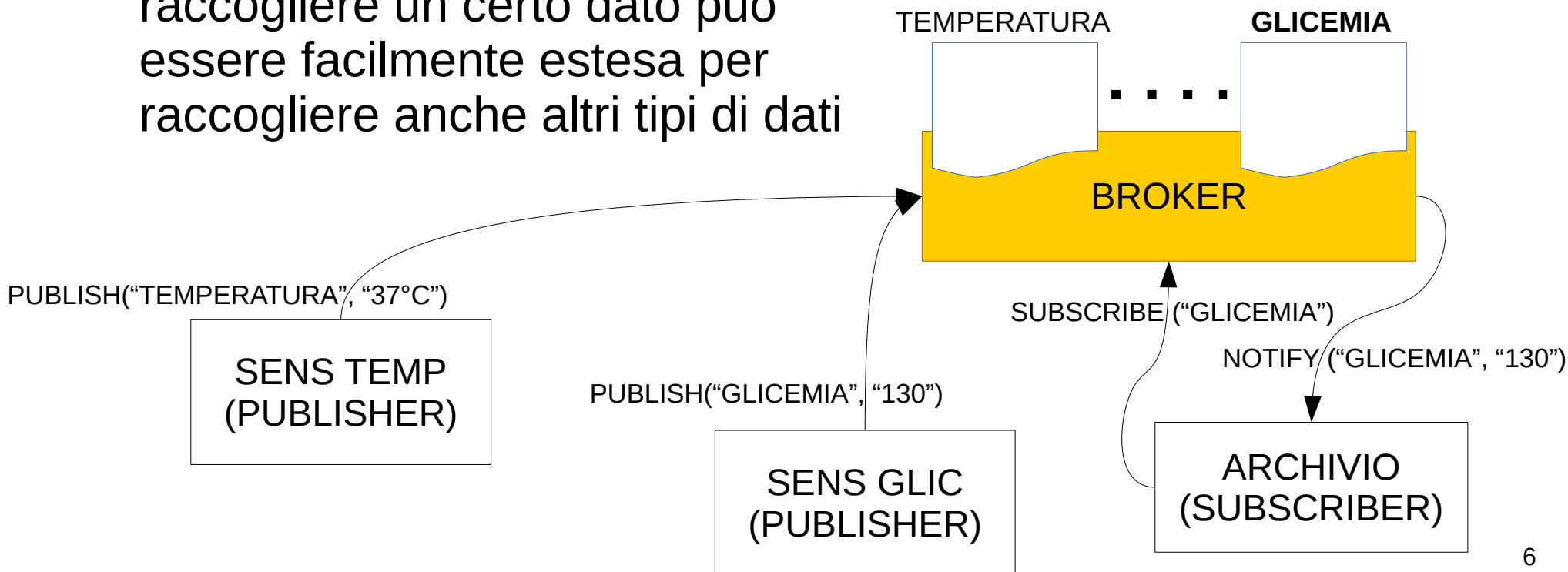
Vantaggi dell'architettura Pub/Sub

- Architettura multi-a-molti



Vantaggi dell'architettura Pub/Sub

- Un'architettura costruita per raccogliere un certo dato può essere facilmente estesa per raccogliere anche altri tipi di dati



Confronto tra Pub/Sub e Client/Server

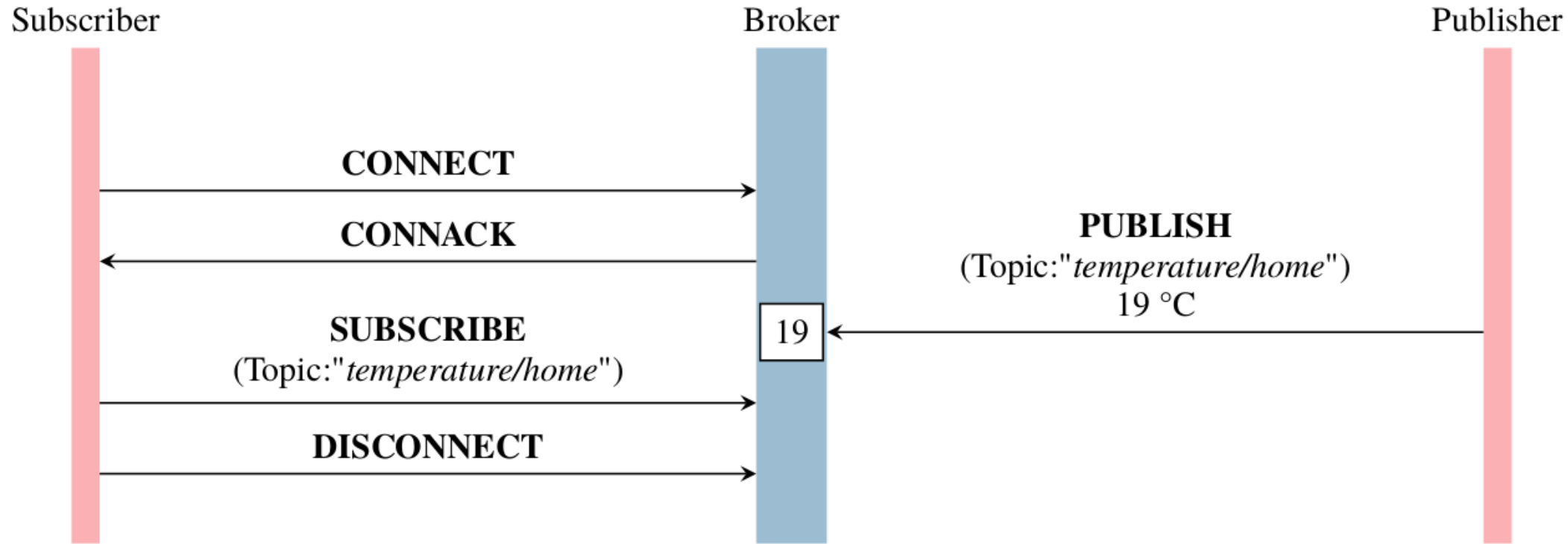
- Pub/Sub è “data centric” mentre C/S è “host centric”
 - In Pub/Sub ciò che viene evidenziato è chi sono i produttori e consumatori di un certo tipo di informazione
 - Produttori e consumatori non interagiscono direttamente
- Un'applicazione basata su Pub/Sub è più facilmente estendibile nel
 - variare il numero di produttori e consumatori di informazione
 - aggiungere tipi diversi di informazione trattati
- Pub/Sub è più leggero a livello di byte trasmessi del più usato approccio C/S che è l'HTTP.

Esempi di protocolli/ambienti per applicazioni Pub/Sub

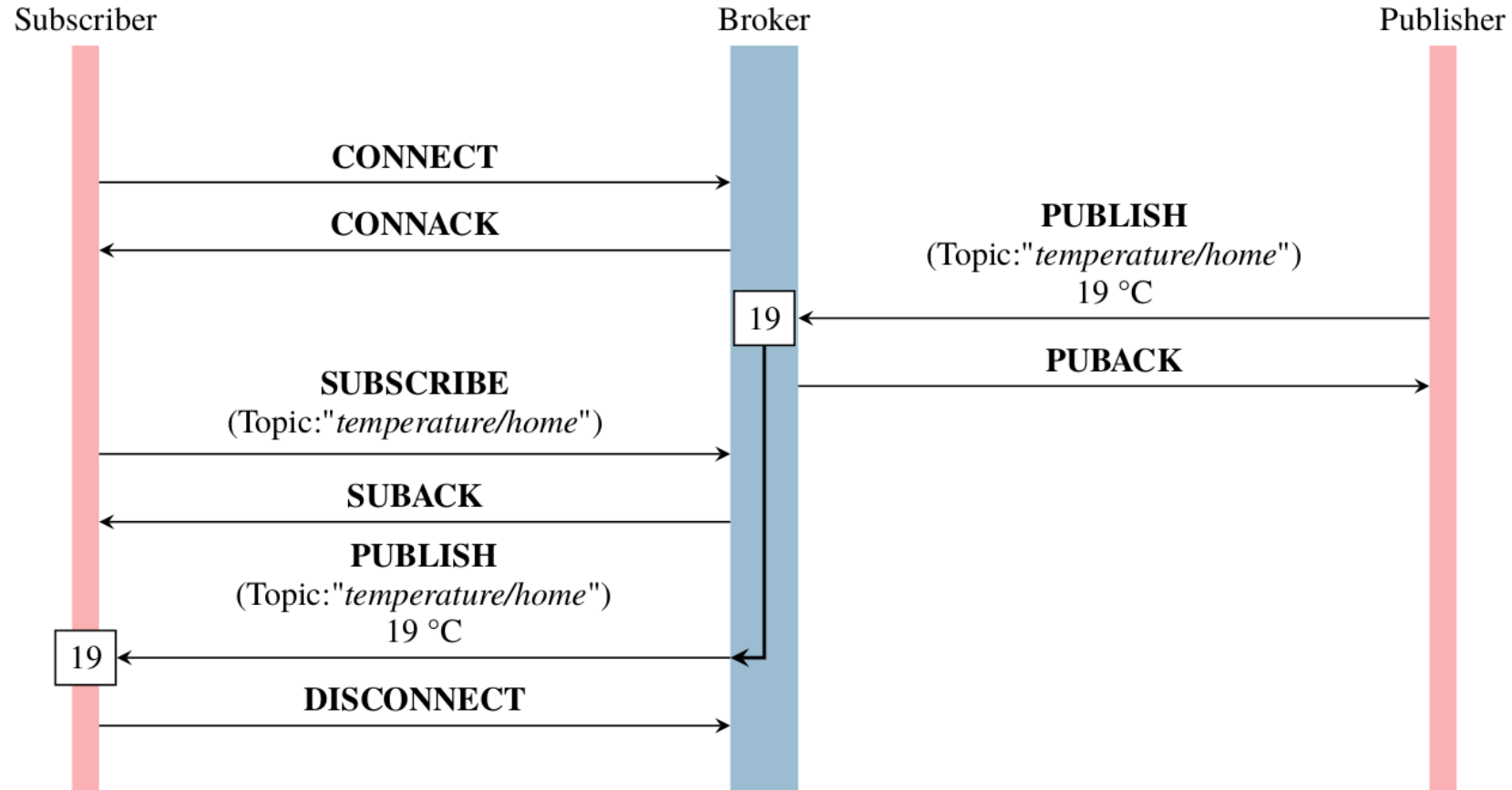
- Message Queuing Telemetry Transport (MQTT)
 - <https://mqtt.org/>
 - Implementazioni
 - Mosquitto (C/C++)
 - Paho (Python)
- Advanced Message Queuing Protocol (AMQP)
 - RabbitMQ
 - Presenza di code in uscita verso i subscriber per **ricezione retroattiva** e gestione della **priorità**
- Apache Kafka
 - Piattaforma distribuita di gestione di **stream di eventi**
 - <https://kafka.apache.org/>

Message Queuing Telemetry Transport (MQTT)

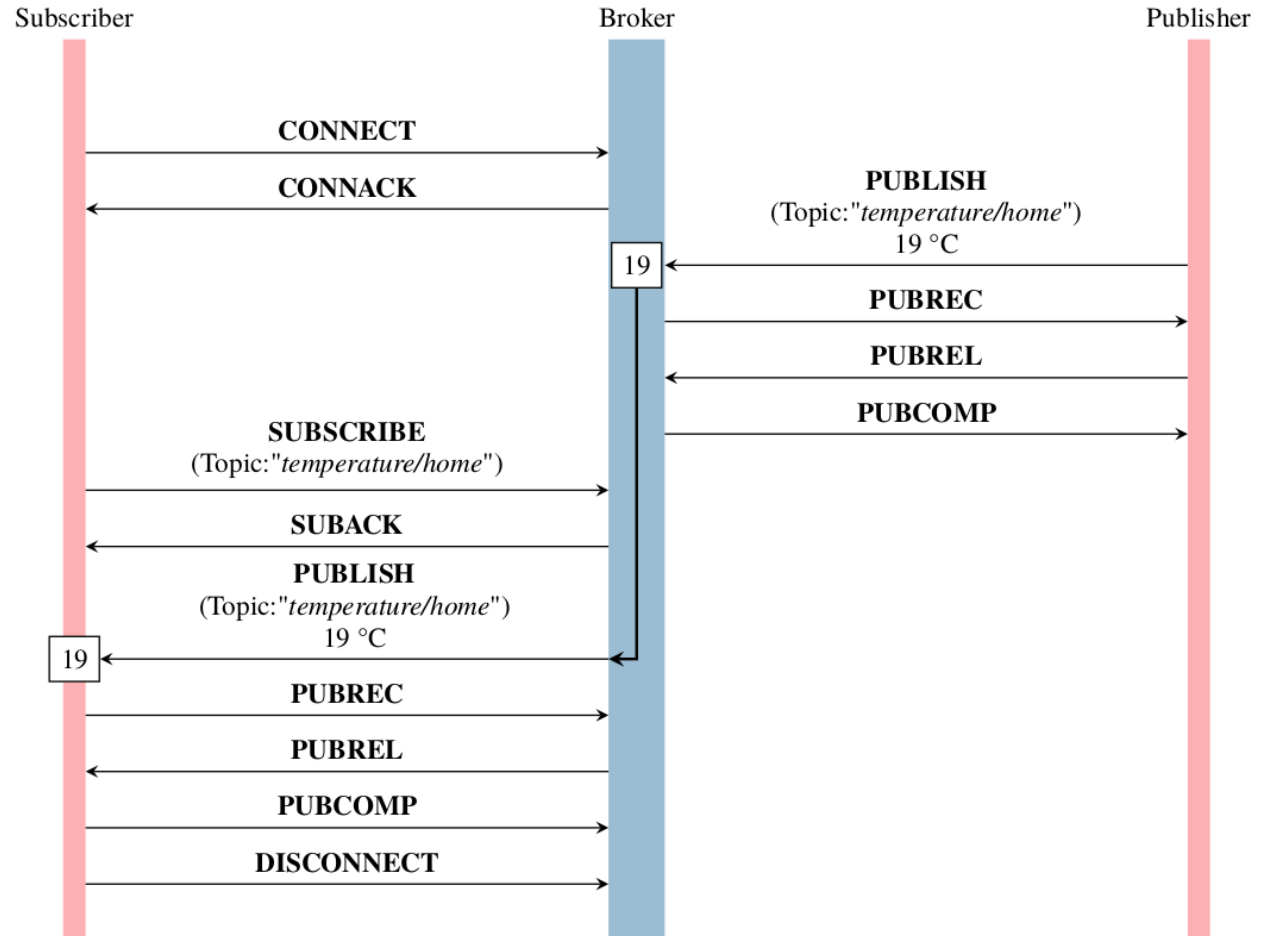
MQTT: QoS Livello 0 (at most one)



MQTT: QoS Livello 1 (at least one)



MQTT: QoS Livello 2 (exactly one)



MQTT: Struttura del topic

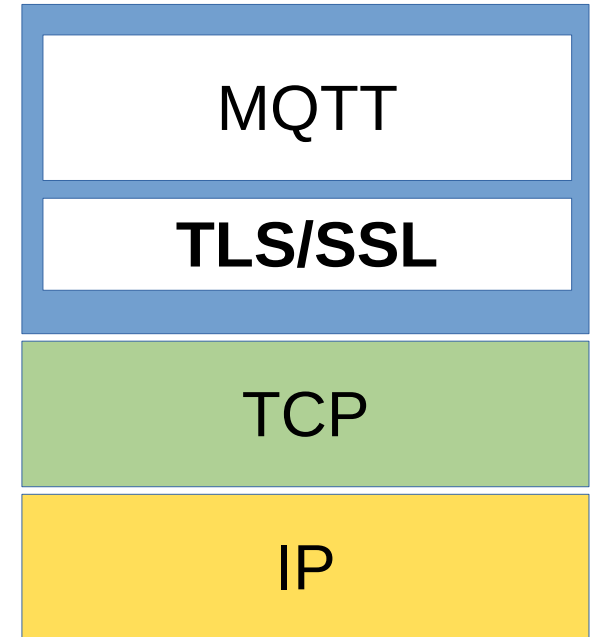
- Servono ad etichettare i dati per indicarne la tipologia
- Stringhe di testo (si sconsiglia l'uso di spazi, meglio "-" o "_")
- Possibilità di utilizzare il carattere "/" per creare gerarchie di argomenti
 - casa/sensori/temperatura/cucina
 - casa/sensori/temperatura/soggiorno
- Ciò che è compreso tra "/" si chiama "livello"
 - Mai iniziare o terminare con "/"
- Possibilità per i subscriber di usare i caratteri "#" e "+" come sostituti di parti di topic (wildcards)

Wildcards

- Uso di “#”
 - Permette di sottoscrivere a tutti i topic che si ottengono sostituendo # con una sequenza di livelli DI SUFFISSO reali
 - Es:
 - Invece di sottoscrivere al topic level1/level2/level3 e level1/level2/level3bis basta sottoscrivere a level1/level2/#
- Uso di “+”
 - Permette di sottoscrivere a tutti i topic che si ottengono sostituendo + con una sequenza di livelli DI PREFISSO reali
 - Es:
 - Invece di sottoscrivere al topic level1/level2/level3 e level1bis/level2/level3 basta sottoscrivere a +/level2/level3

Broker

- E' l'unico attore della famiglia MQTT che implementa connessioni TCP lato server
 - Porta 1883 per trasmissioni in chiaro
 - Porta 8883 per trasmissioni TCP con sopra TLS/SSL (serve impostare certificato e chiave privata)
- Broker, publisher e subscriber possono stare su sistemi hardware di tipo diverso (Intel, ARM, RISC-V), sistemi operativi diversi ed essere scritti con linguaggi di programmazione diversi
 - È il protocollo MQTT che mette tutti d'accordo



ATTENZIONE

- La presenza del broker sulle proprie macchine è un rischio per la sicurezza!
 - Per questo motivo per gli scopi del corso viene messo a disposizione un broker in cloud
 - Sulle macchine del laboratorio avete già tutto quello che serve
 - Sulla propria macchina in Linux potete installare SOLO publisher e subscriber attraverso i seguenti comandi da terminale
 - `sudo apt update`
 - `sudo apt install mosquito-clients`
- Sia in Windows che in MacOS il processo BROKER viene installato insieme a publisher e subscriber (vedi documento di istruzioni contenuto nello ZIP) e attivato automaticamente all'avvio del sistema
 - Per ragioni di sicurezza è bene che il BROKER non resti attivo inutilmente per lunghi periodi di tempo
 - Sul documento di istruzioni è descritto come disattivarlo ogni volta
 - Quando non si usa Mosquitto per lungo tempo è meglio disinstallarlo

Esempio di applicazione Pub/Sub

- Si assume di essere in rete (o in laboratorio o sul proprio PC anche a casa) per poter accedere ad un broker attivo all'indirizzo IP 90.147.167.187
- Estrarre il file "ISRG_Root_X1.pem" dallo ZIP e metterlo nella cartella di lavoro
- Aprire due finestre di terminale e posizionarsi nella propria cartella di lavoro dove c'è il file "ISRG_Root_X1.pem" appena estratto
- Per sottoscrivere ad un topic
 - `mosquitto_sub -h 90.147.167.187 -p 8883 -t temperatura -u univr-studenti -P MQTT-esercitazione2024 --cafile ISRG_Root_X1.pem --insecure`
- Per pubblicare su di un topic
 - `mosquitto_pub -h 90.147.167.187 -p 8883 -d -t temperatura -m "25" -u univr-studenti -P MQTT-esercitazione2024 --cafile ISRG_Root_X1.pem --insecure`
- **ATTENZIONE** che in Windows i nomi dei file eseguibili del subscriber e del publisher terminano con ".exe"

Descrizione dei parametri sulla linea di comando

- h → host a cui connettersi
- p (da notare che è minuscola) → porta
- d → abilita la modalità debug (può essere utile per vedere tutto quello che fa il client, quindi anche la connessione)
- t → nome del topic
- m “messaggio” → messaggio da pubblicare per quel topic
- u → username
- P (da notare che è maiuscola) → password
- cafile → certificato pubblico per verifica del server nella connessione TLS
- insecure → da usare se in -h si mette l'indirizzo IP e non il dominio

Esercizi su pub/sub

- Partendo dall'esercizio mostrato nelle istruzioni, cosa succede se pubblico una temperatura prima di aver lanciato il subscriber?
- Partendo dall'esercizio mostrato nelle istruzioni creare un'applicazione pub/sub con 2 sensori di temperatura relativi a 2 stanze diverse. Quante finestre di terminale devo aprire?
- Partendo dall'esercizio precedente come fare per avere un unico subscriber per entrambe le temperature? Come si fa a distinguere da quale stanza proviene la temperatura?
- Prova a pubblicare un valore di umidità relativa (topic "UR"); il subscriber interessato alle temperature lo riceve? Come si fa a creare un subscriber interessato all'umidità? Costruire un'applicazione pub/sub con 4 finestre per produrre e visualizzare sia valori di temperatura sia valori di umidità.
- Aprire e studiare con Wireshark il file PCAP contenuto nello ZIP
 - Quale protocollo di livello trasporto utilizza MQTT? Quali sono le porte?
 - Trovare le fasi di publish e subscribe. Quante connessioni TCP apre un subscriber? Quante connessioni TCP apre un publisher?

Esercizi su pub/sub

- Si vuole costruire con MQTT un servizio di messaggistica universitaria
 - Il rettore può leggere tutti i messaggi
 - La segreteria può leggere i messaggi dai docenti e dagli studenti
 - I docenti possono leggere i messaggi dai docenti e dagli studenti
 - Gli studenti possono leggere solo i messaggi degli altri studenti

ATTENZIONE

- Il processo BROKER deve essere attivo durante gli esercizi con applicazioni pub/sub
- Sia in Windows che in Linux il processo BROKER viene attivato automaticamente all'avvio del sistema
- Per ragioni di sicurezza è bene che il BROKER non resti attivo inutilmente per lunghi periodi di tempo
 - Sul documento di istruzioni è descritto come disattivarlo ogni volta
 - Quando non si usa Mosquitto per lungo tempo è meglio disinstallarlo