Internet of Things
IO 4041
Application Layer Protocols

IoT Protocols

Application Layer: HTTP, CoAP, WebSockets, MQTT, XMPP, DDS, AMQP, …

Transport Layer: TCP, UDP

Network Layer: Ipv4, IPv6, 6 LoWPAN

Link Layer: 802.3 Ethernet, 802.11 WiFi, 802.16 WiMax, 802.15.4 LOWPAN, 2G/3G/LTE/Cellular

Application Layer Protocols

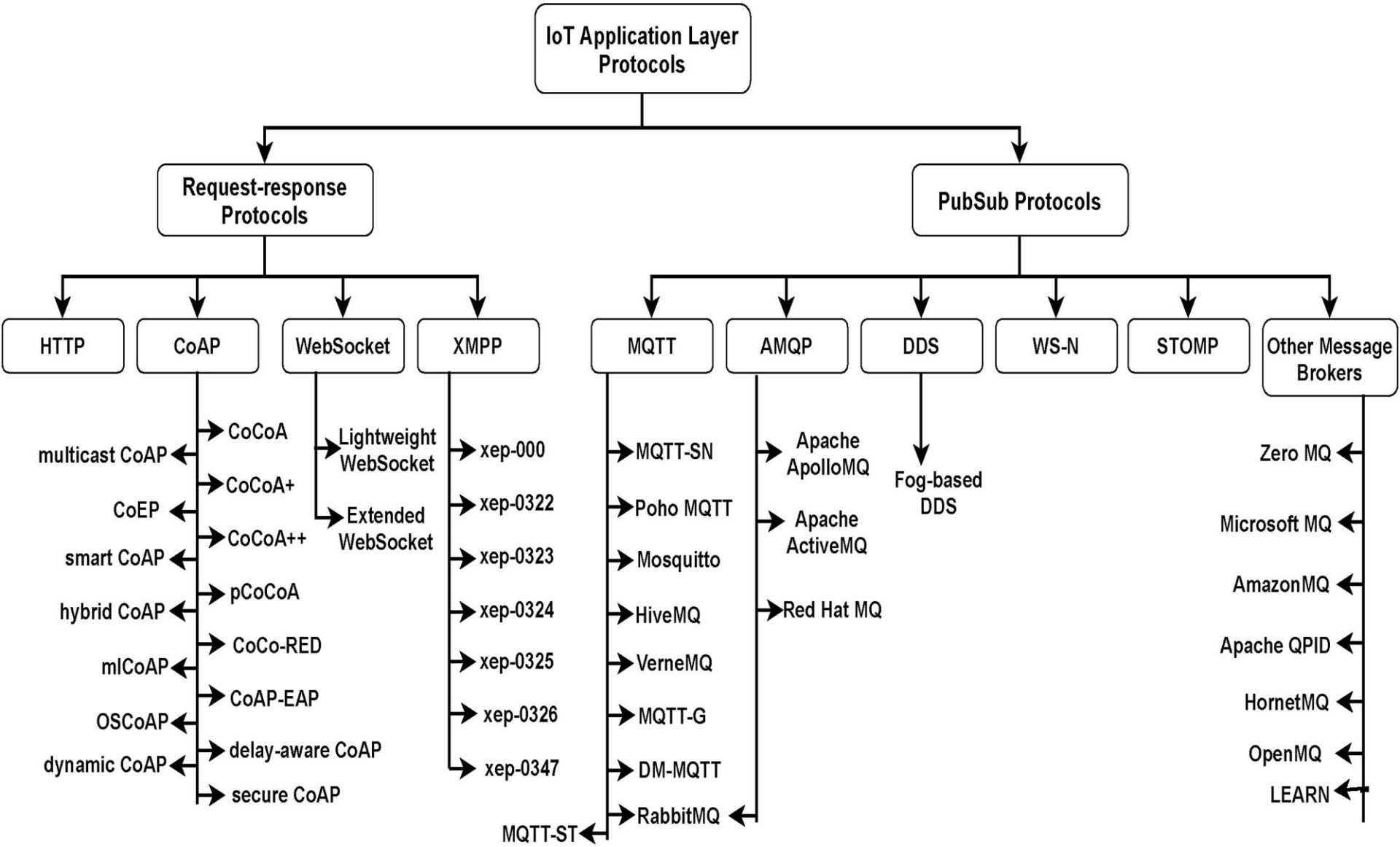
Why we need more protocols for application layer, when already there are so many protocols in the internet world.
HTTP: browsers are HTTP clients.
Protocol for moving files (FTP, and the SFTP)
So, many more protocols are around there.
Why one more why not adopt existing ones and use them,
why create many more that may bother us.

Application Layer Protocols

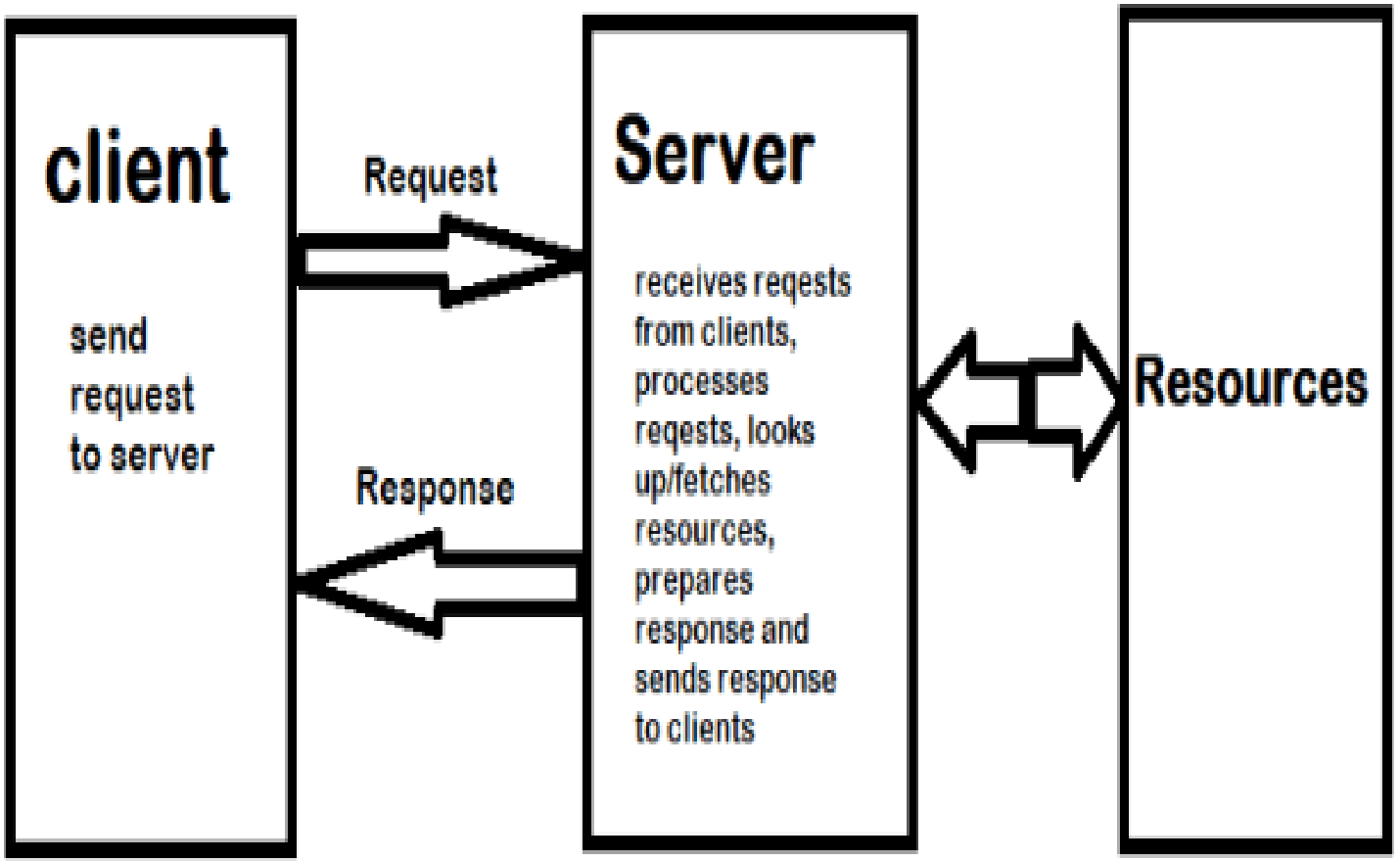
Based on the type of device involved and the function it will perform,
the right protocol for a particular IoT application depends on a number of factors such as

- **Data latency**: how fast data transport is? Time for a packet from one end to another
- **Reliability**
- **Bandwidth**: volume of data needs to be accommodated?
- **Transport**: best transport protocol for IoT?

Application Layer Protocols

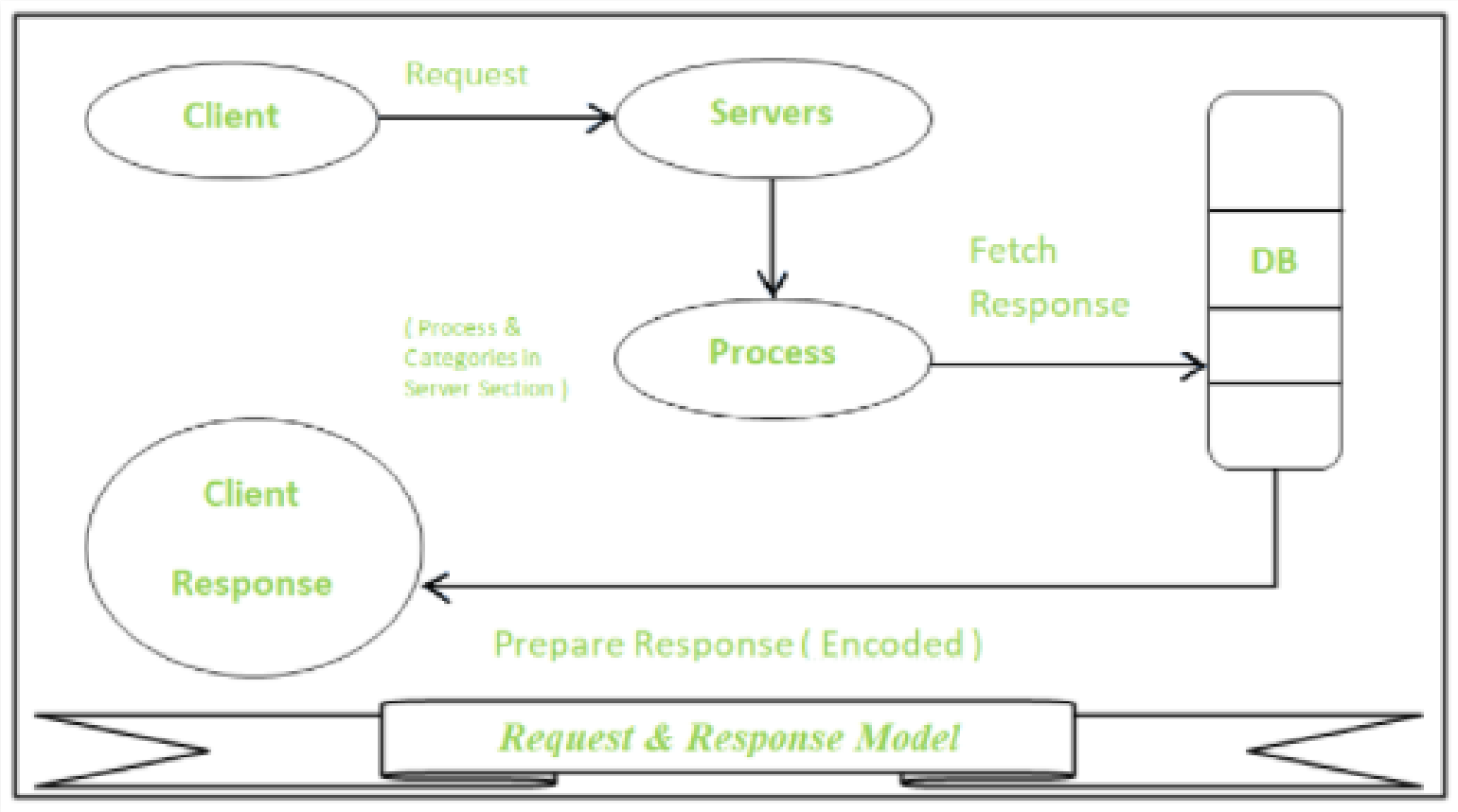


Request Response Model



Request-Response Communication Model

Request Response Model



Request Response Model

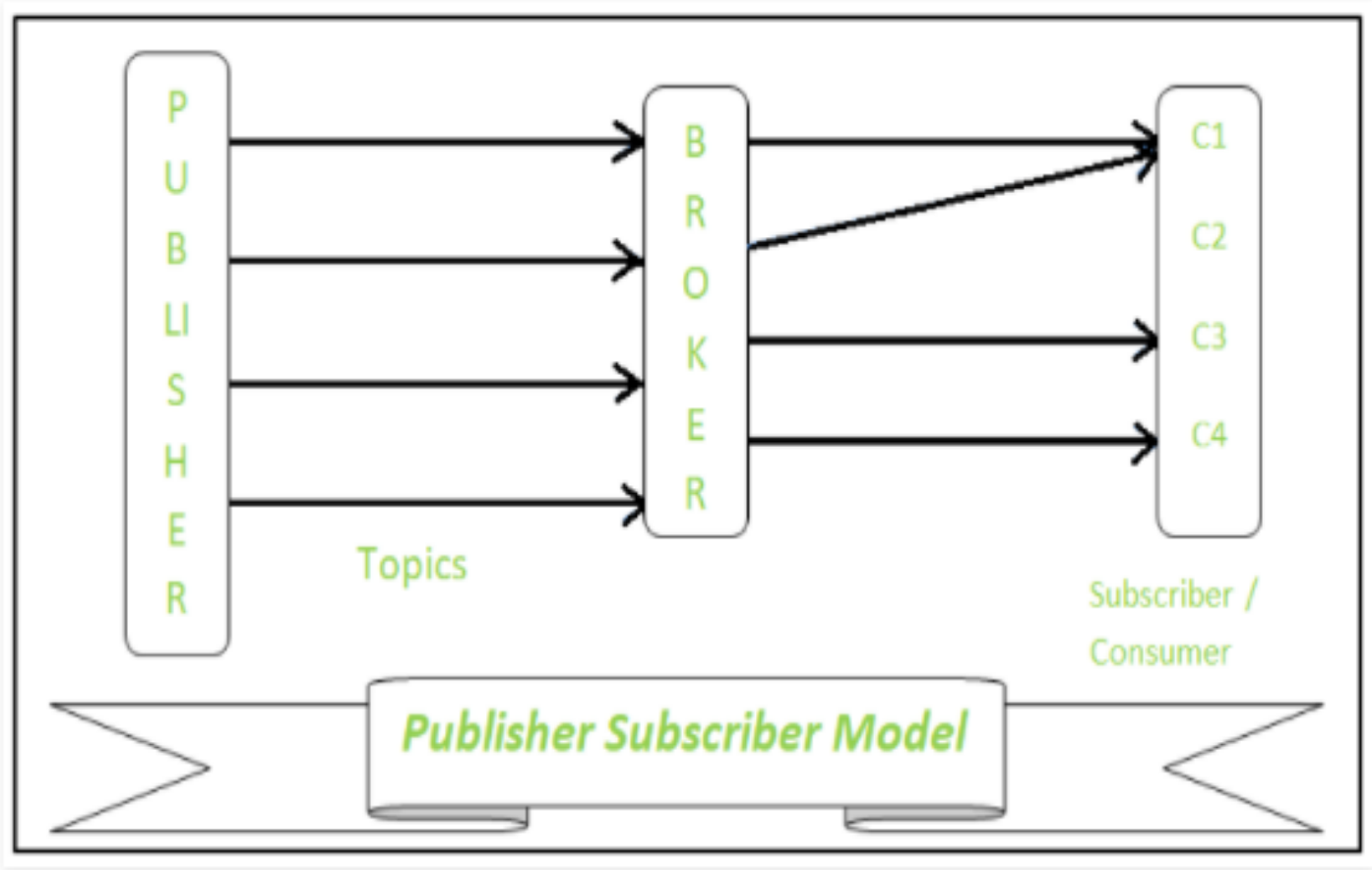
- Client: an IoT device sends a request to server (for data transfer or upload)
- Server responds to the request

- Stateless model and each request is independently handled

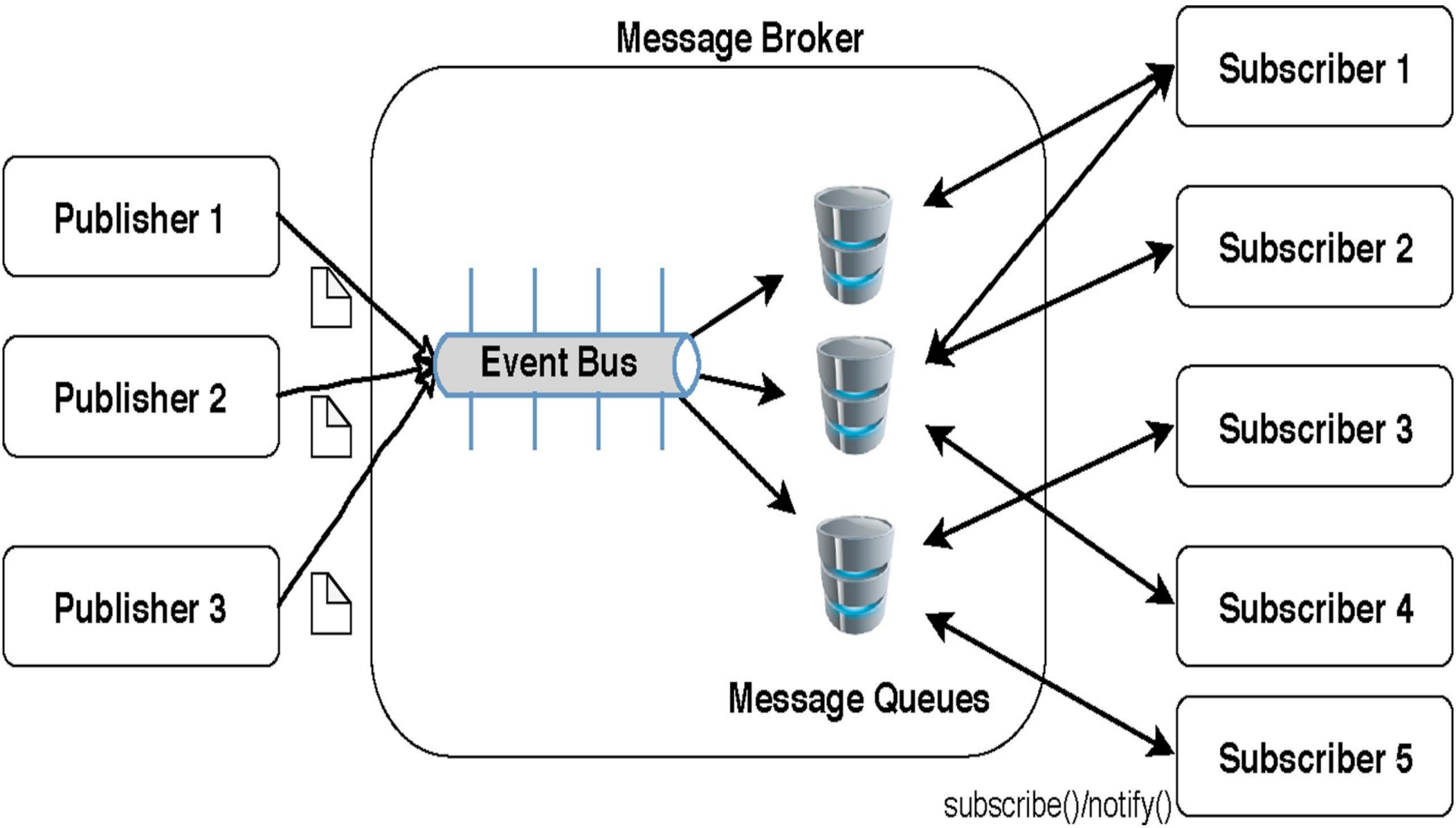
- Server can handle request from multiple clients

- When the server receives the request
  - it decides how to respond,
  - fetches the data retrieves resources, and
  - prepares the response, and sends it to the client.

Publish–Subscribe Model



Publish–Subscribe Model



Publish–Subscribe Model

- 3 components: Publishers, Brokers, and Subscribers/Consumers
- **Publishers** (source of data). It sends the data to the topic which are managed by the broker. They are not aware of consumers.
- **Subscribers/Consumers** subscribe to the topics which are managed by the broker.
- **Brokers** accept data from publishers and send it to the appropriate subscribers.
  - has the information regarding the subscriber to a particular topic

Publish–Subscribe Model

- Endpoints do not know about each other, but the broker knows about them
- The broker makes a bridge between publishers and subscribers during data transmissions

Message Queuing Telemetry Transport (MQTT)

MQTT

- Andy Stanford-Clark and Arlen Nipper initially developed MQTT at IBM in 1999,
- It was standardized in 2013 by the Organization for the Advancement of Structured Information Standards (OASIS).
- lightweight PubSub messaging protocol widely used by many web applications, including the IoT
- MQTT connection uses M2M communication with a many-to-many routing mechanism

MQTT

- Introduced by IBM in 1999
- Extremely Lightweight Publish subscriber model
- Components
  - Publisher
  - Subscriber
  - Brooker: to connect publisher and subscriber
    - Topics: Data classification
- Methods
  - connect and disconnect
  - publish and subscribe

MQTT

Topics: temperature of the hall
home/hall/temperature (a particular place)

- Publisher publish on this topic
- Only subscriber of this topic get data

MQTT: wildcard

Topics: Temperature of various places
home/+/temperature (single level wild card)

Arbitrary values of allowed in place of + (hall, kitchen, bed room etc)
A subscriber can subscribe by using single level wild card

Single-level wildcards "+" can appear anywhere in the topic string

Cannot be used wildcards when publishing

MQTT

Topics: various aspects of home
home/# (multi level wild card)

A subscriber can subscribe to any topic starting with home

Multi-level wildcards "#" must appear at the end of the string

MQTT

For same topic

- One to one: one publisher one subscriber
- One to many: single publisher many subscribers
- Many to one: many publishers single subscriber

A message server / broker matches publications to subscriptions

• If no matches, the message is discarded

• If one or more matches the message is delivered to each matching subscriber/consumer

MQTT

A subscription can be durable or non durable

Durable: Once a subscription is in place a broker will forward matching messages to the subscriber:

- Immediately if the subscriber is connected
- If the subscriber is not connected, messages are stored on the server/broker until the next time the subscriber connects

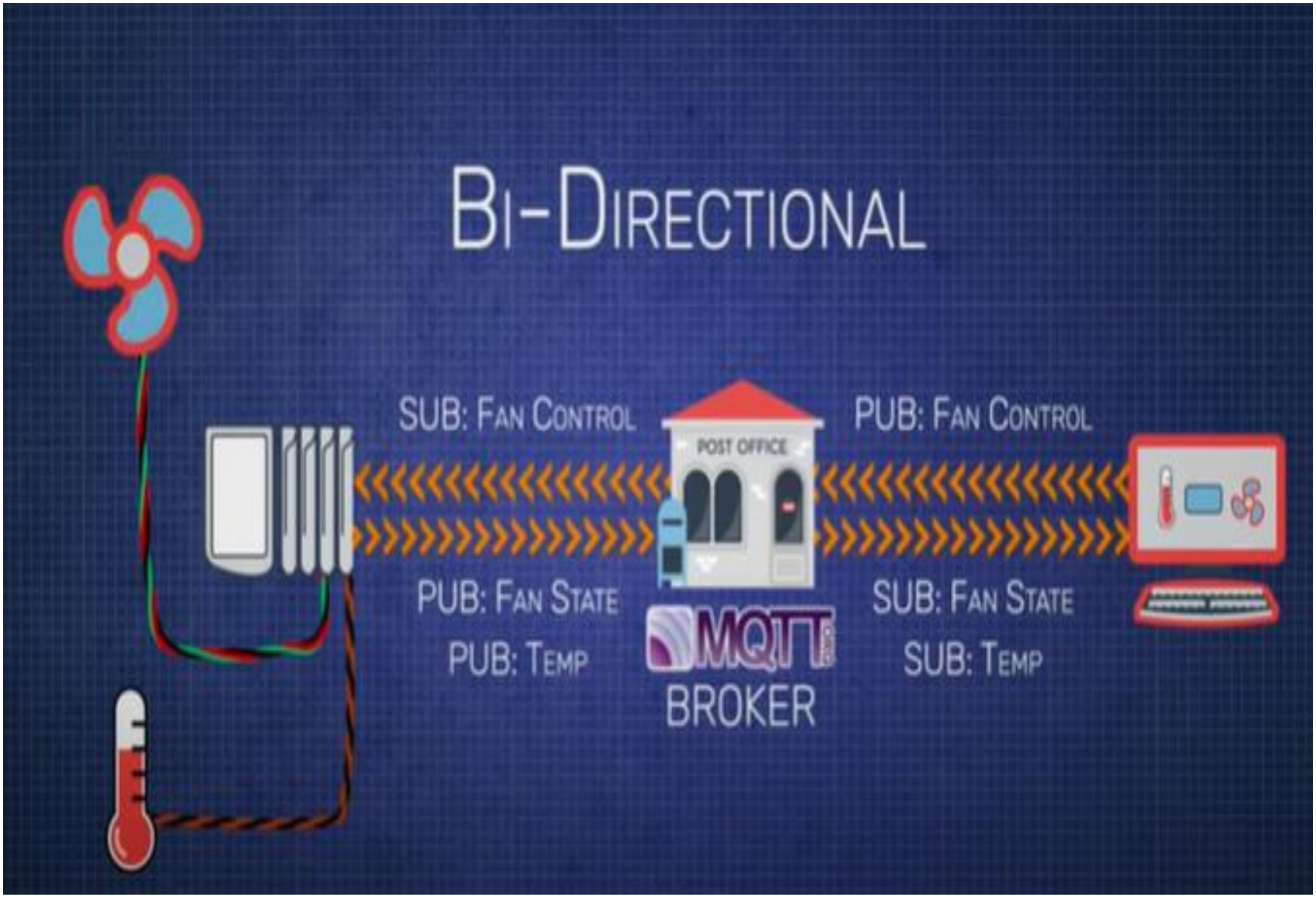
Non-durable: The subscription lifetime is the same as the time the subscriber is connected to the server / broker

MQTT

A publication may be retained

- A publisher can mark a publication as retained
- The broker / server remembers the last known good message of a retained topic
- The broker / server gives the last known good message to new subscribers
  - i.e. the new subscriber does not have to wait for a publisher to publish a message in order to receive its first message

MQTT: **Bidirectional**



MQTT: Broker

Store data in the form of retained messages

- New subscriber to the topic can get last value straightway
- **Example**: low power sensor publishes after every 6 hours
- Broker subscribes to database clients
- **Persistent sessions:** Broker keeps track of each session and can even store session information long term as devices go on and off line