GNUnet MQTT

Distributed M2M Communication

We (will) live in a M2M world

- In 2013, 39% = 2.2bn people with Internet access
- ~7.7bn people by 2020
  - ~3bn with Internet Access
- >30bn Internet Connected Devices by 2020
- **avg. 10 Internet Devices / person!**

We (will) live in a M2M world

We (will) live in a M2M world

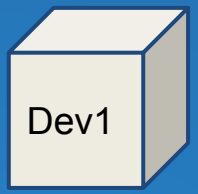
We (will) live in a M2M world

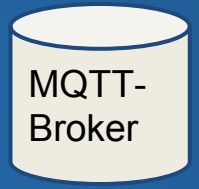
We (will) live in a M2M world

How to connect 30bn devices?

Messaging	• CoAP • MQTT, AMQP, … • Websockets • …
Transport	TCP, UDP
Network	IPv6
Physical + Data Link	ZigBee, IEEE 802.15.4, WiFi

MQTT

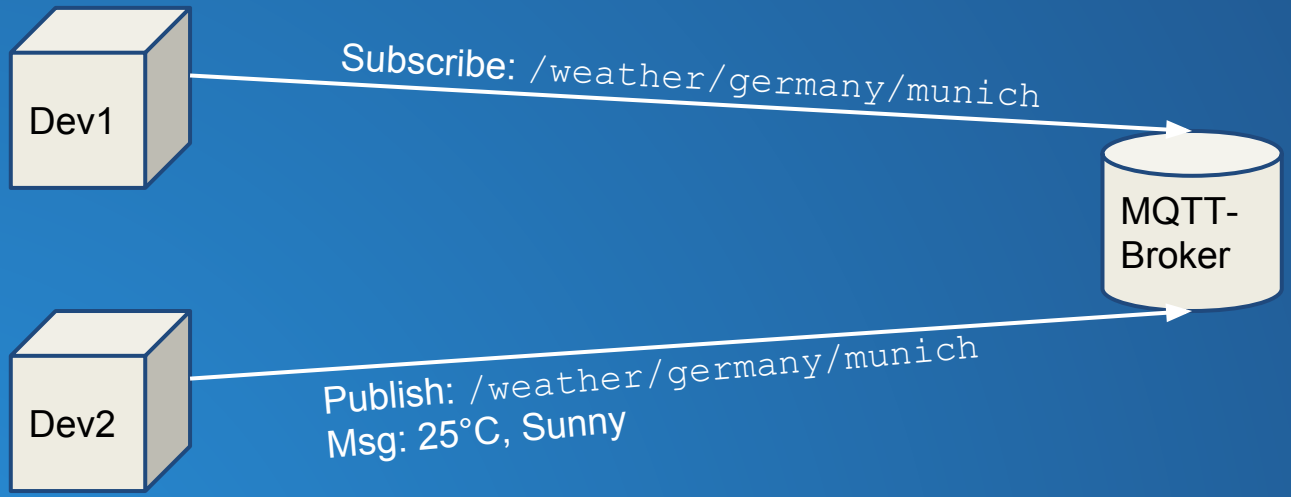




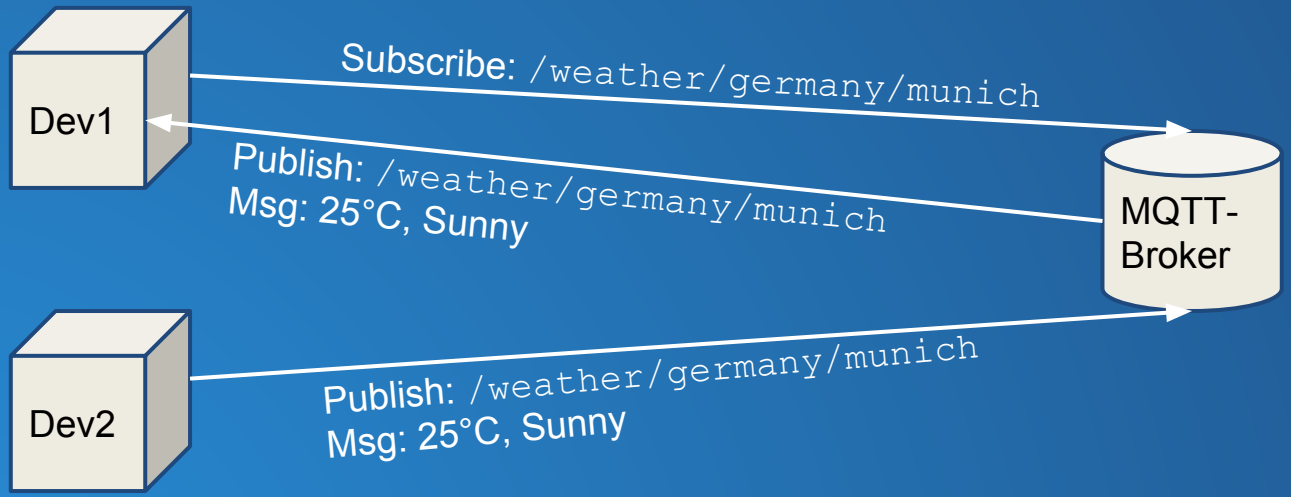
MQTT

MQTT

MQTT


Dev1
Subscribe: /weather/germany/munich
MQTT-
Broker
Dev2
Publish: /weather/germany/munich
Msg: 25°C, Sunny

MQTT


Dev1
Subscribe: /weather/germany/munich
Publish: /weather/germany/munich
Msg: 25°C, Sunny
MQTT-
Broker
Dev2
Publish: /weather/germany/munich
Msg: 25°C, Sunny

Cool MQTT Features

- Wildcard Matching

Wildcard Matching

- +
  - Match exactly one level
  - `/weather/+/munich` matches all cities called munich such as Munich, North Dakota
- *
  - Match all sublevels
  - `/weather/germany/*` matches all german cities and their districts e.g. Sendling

Cool MQTT Features

- Wildcard Matching
- Quality of Service

Quality of Service

- QoS0 - Best Effort
- QoS1 - At least once
- QoS2 - At most once per Subscriber

Broker ensures delivery of QoS1 and 2 messages to connected clients

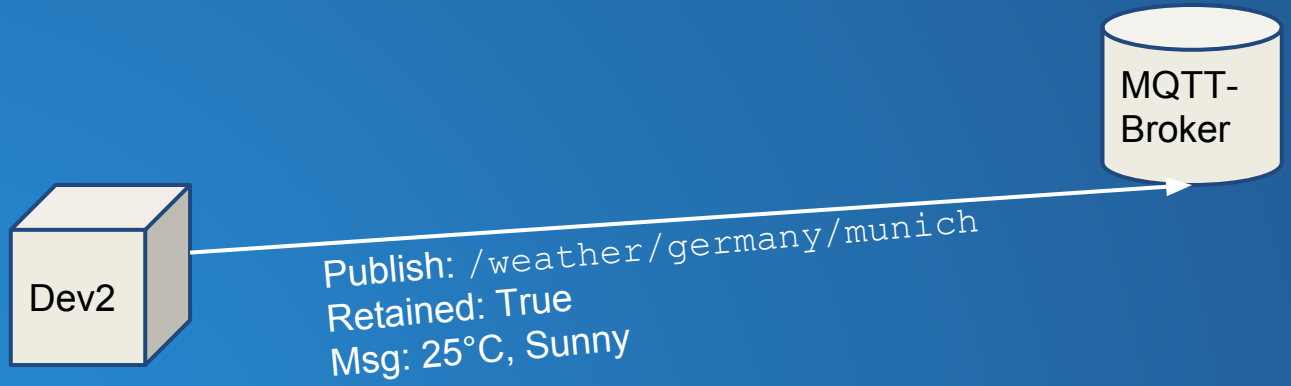
Cool MQTT Features

- Wildcard Matching
- Quality of Service
- Sessions
  - Retained Messages
  - Last Will

Sessions

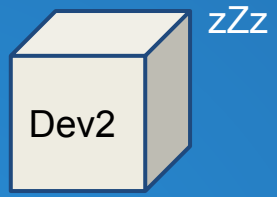
- Retained Messages
  - Broker sends cached messages to new clients

Retained Messages

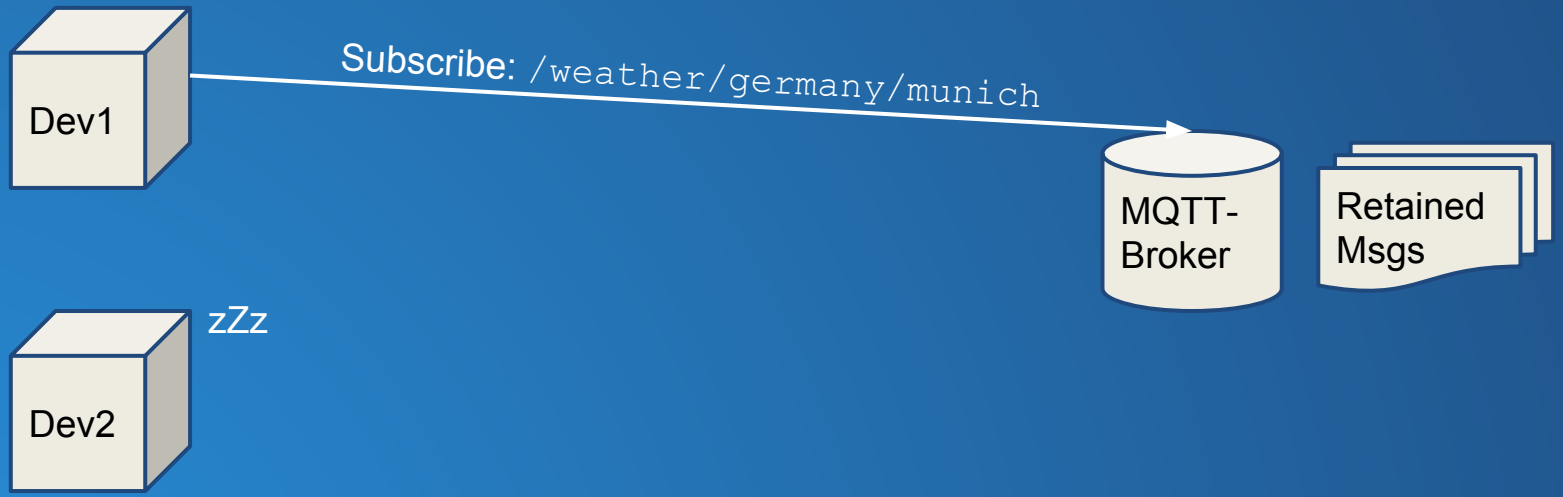


Retained Messages

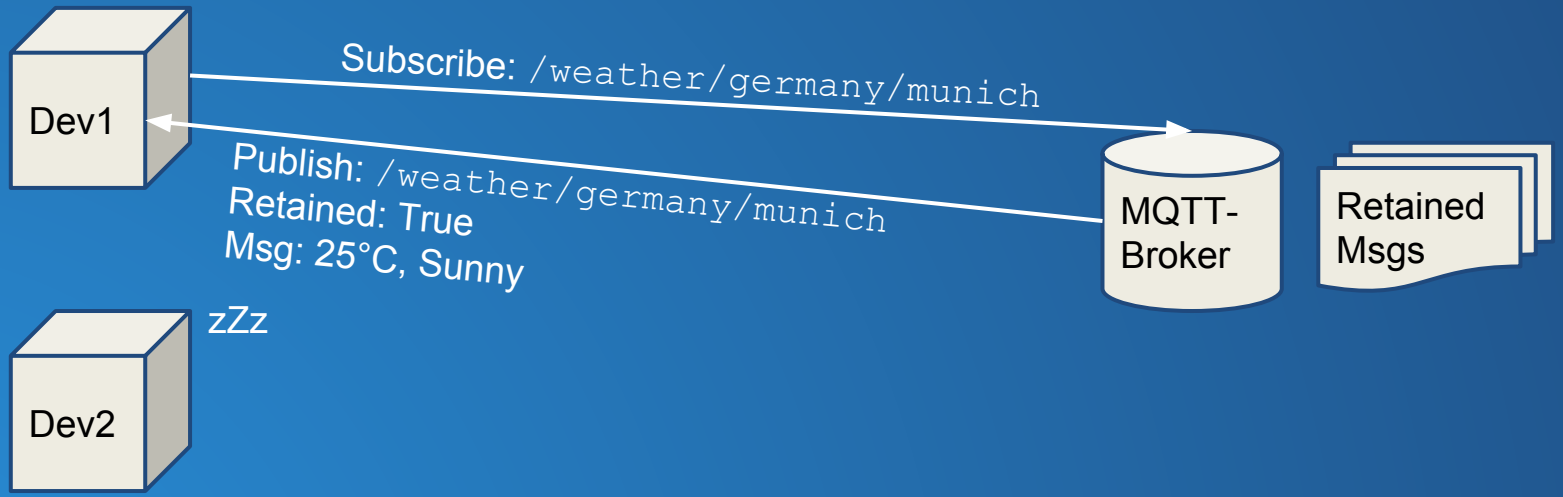
Retained Messages



Retained Messages



Retained Messages



Sessions

- Retained Messages
  - Broker sends cached messages to new clients
- Last Will
  - Broker sends message on behalf of the client on unexpected disconnect

Limitations of MQTT

Limitations of MQTT

- Single point of failure
- Scaling depends on Broker Performance
- Central point of control
- Honeypot to steal data
- No Self-Organisation

MQTT in P2P Network

MQTT

Single Broker – client-server paradigm

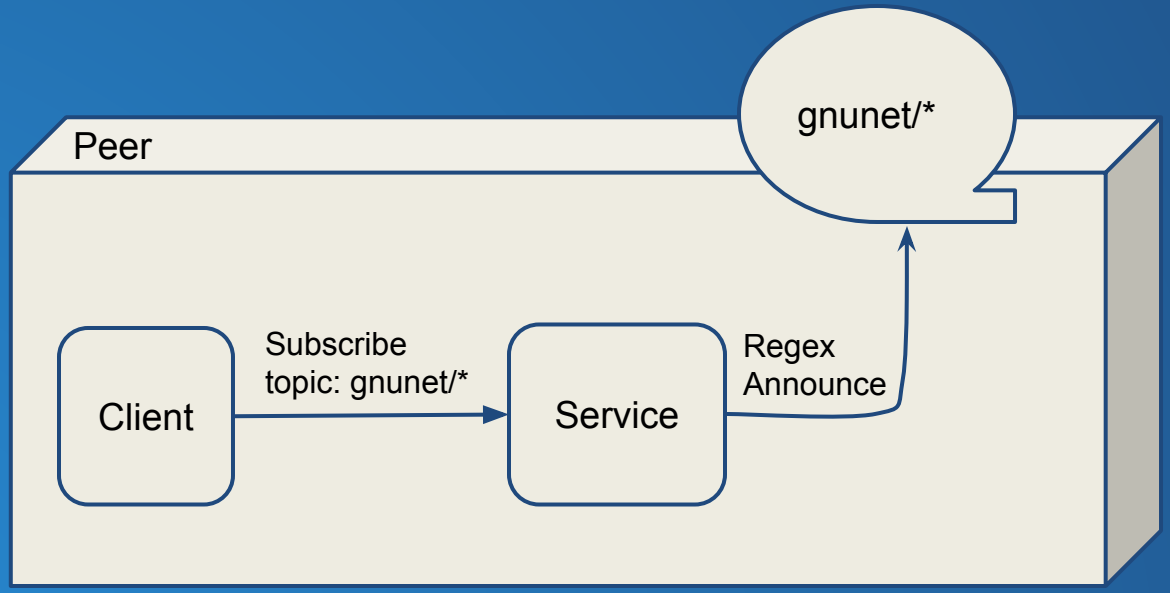
GNUnet MQTT

Every Peer is a broker

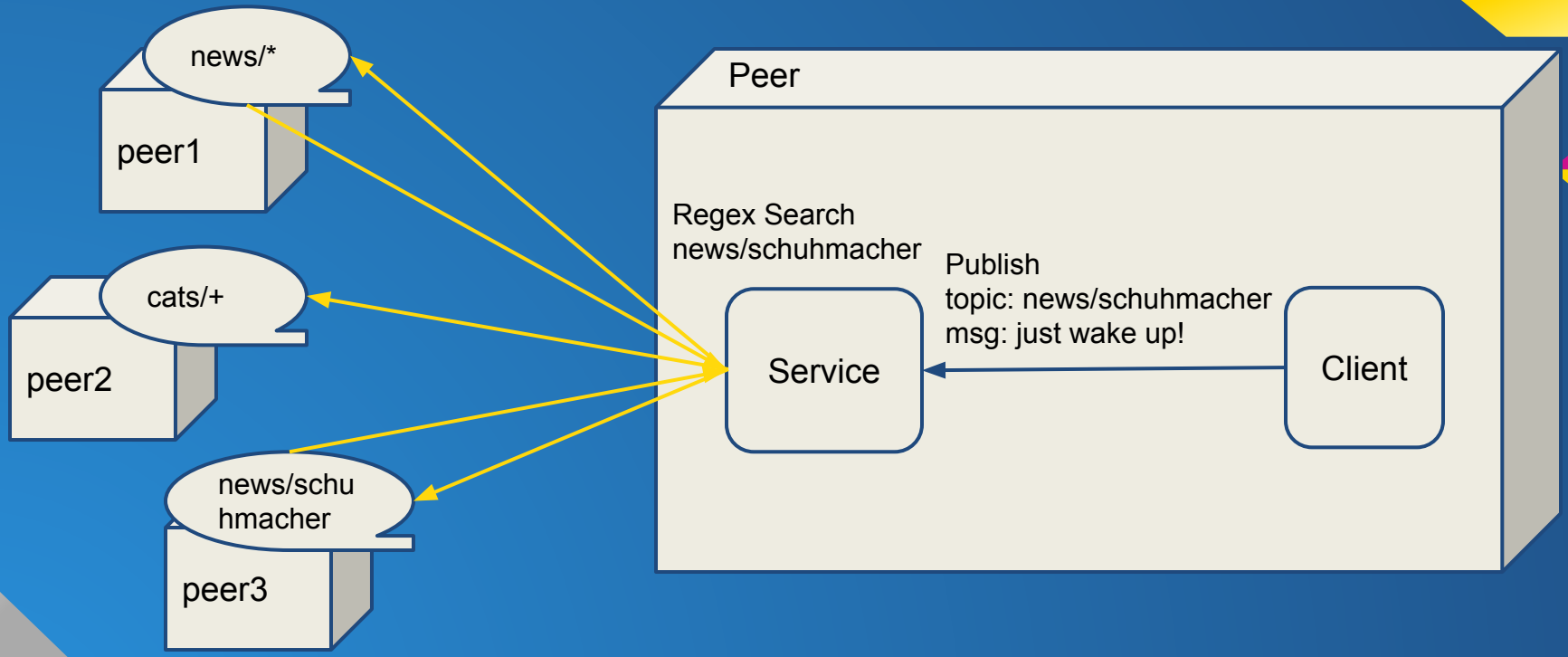
How does it work?

Subscribe	1. Build Regex for Topic 2. ‘Subscribe’ it via `GNUNET_REGEX_announce()`
Publish	1. Search for Peers interested in topic via `GNUNET_REGEX_search()` 2. Send message to all interested peers via `GNUNET_MESH`
Unsubscribe	1. Take down the Regex via `GNUNET_REGEX_cancel()`

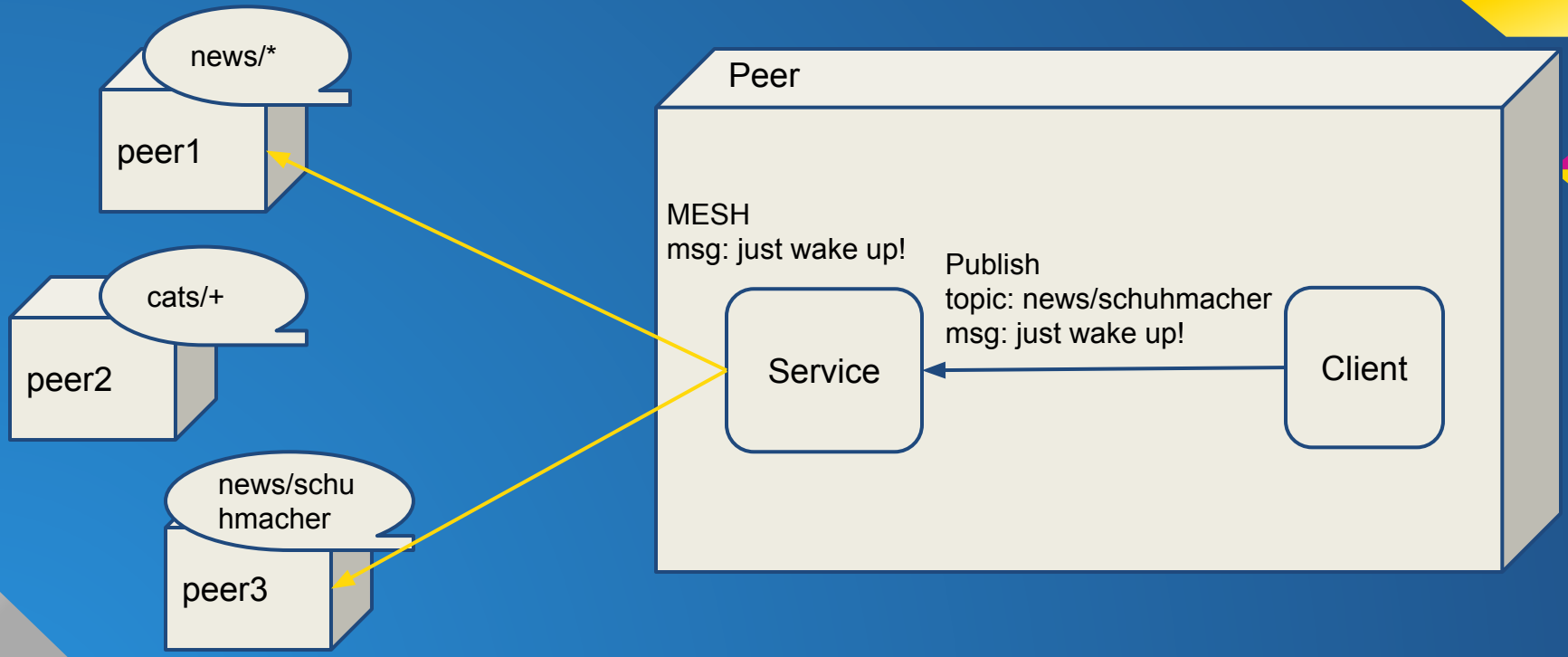
Subscribe


gnunet/*
Peer
Client
Subscribe
topic: gnunet/*
Service
Regex
Announce

Publish


news/*
peer1
cats/+
peer2
news/schu
hmacher
peer3
Peer
Regex Search
news/schuhmacher
Service
Publish
topic: news/schuhmacher
msg: just wake up!
Client

Publish


news/*
peer1
cats/+
peer2
news/schu
hmacher
peer3
Peer
MESH
msg: just wake up!
Publish
topic: news/schuhmacher
msg: just wake up!
Service
Client

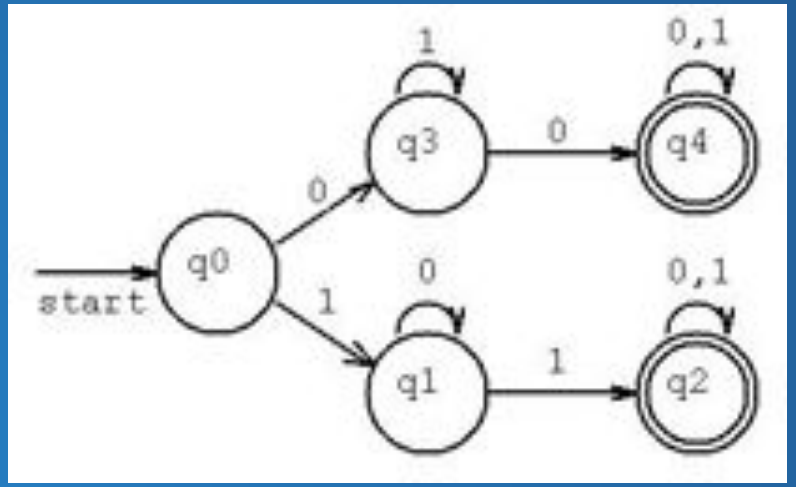
Benefits

- No single Point of Failure
- No single Point of Control
- No Honeypot
- Self-Configuration

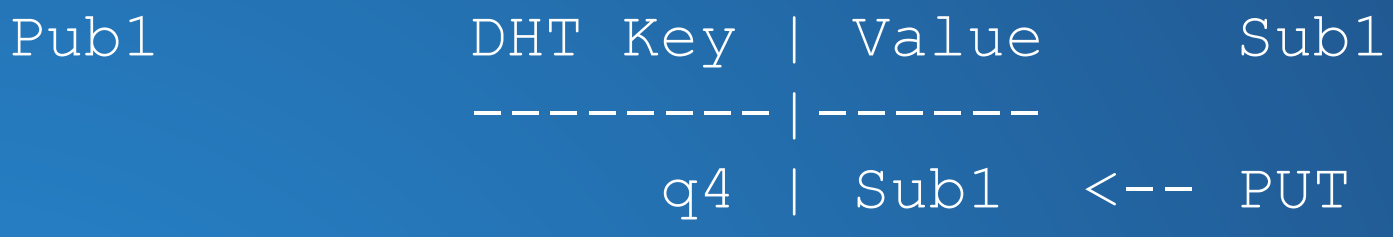
Problems

- We lose consistency
- Increased latency
- Privacy issues
- No access control
  - Any peer can read any topic
  - Any peer can publish on any topic

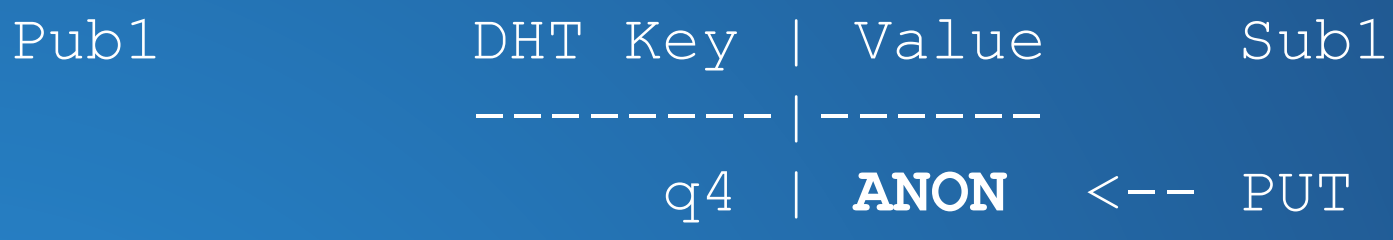
Anonymous Publisher Discovery



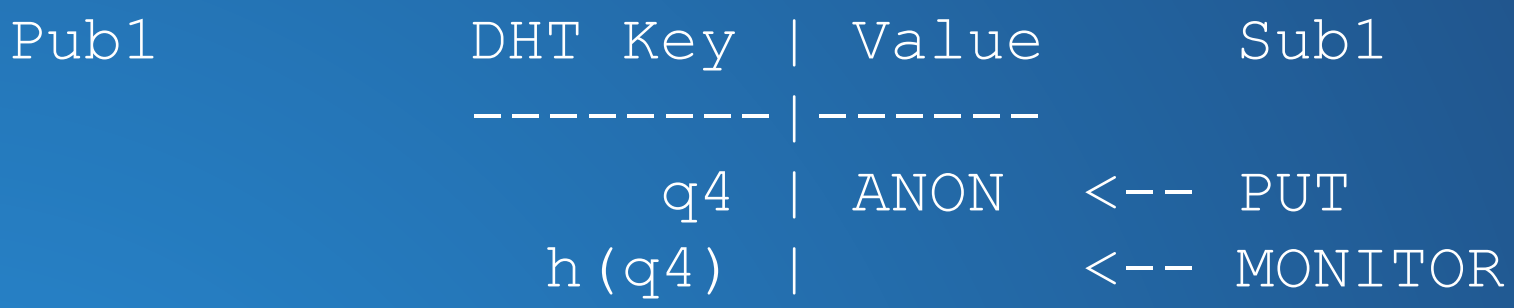
Anonymous Publisher Discovery



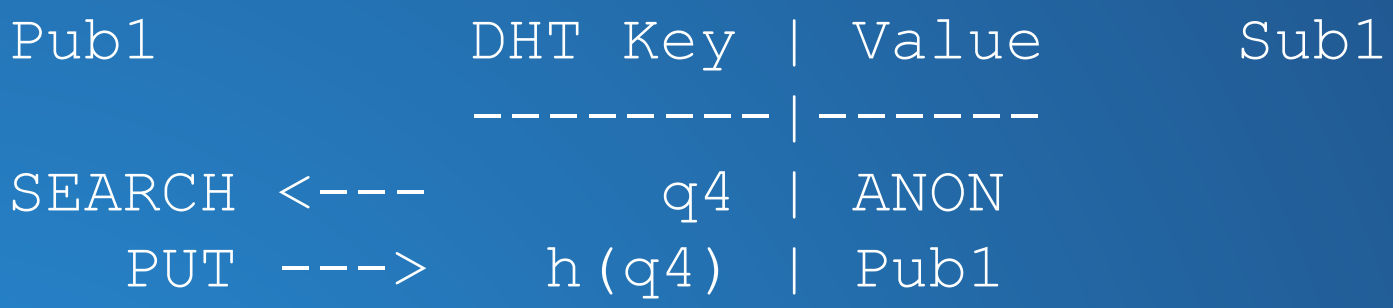
Anonymous Publisher Discovery



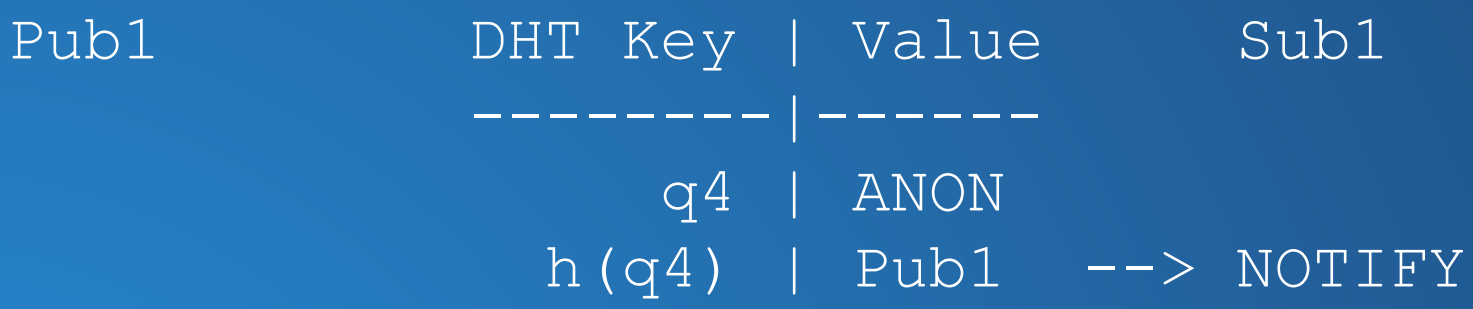
Anonymous Publisher Discovery



Anonymous Publisher Discovery



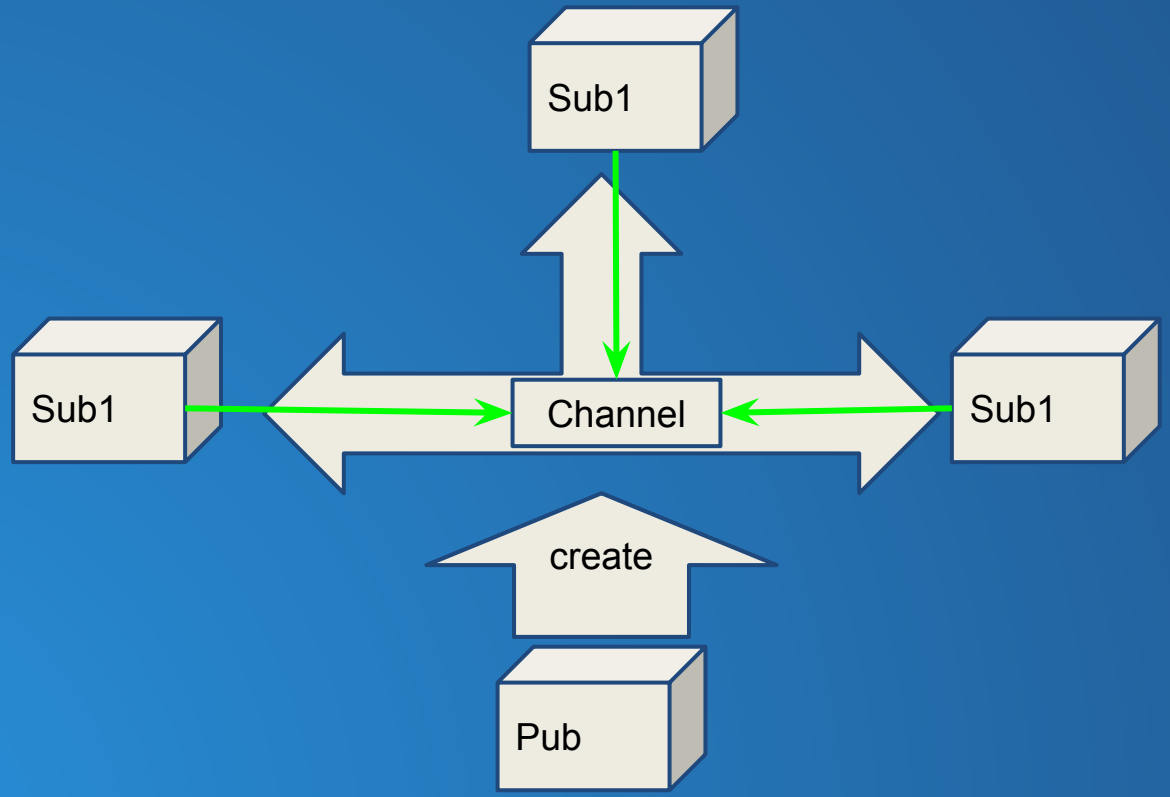
Anonymous Publisher Discovery



Problems

- We lose consistency
- Increased latency
- ~~Privacy issues~~
- No access control
  - Any peer can read any topic
  - Any peer can publish on any topic

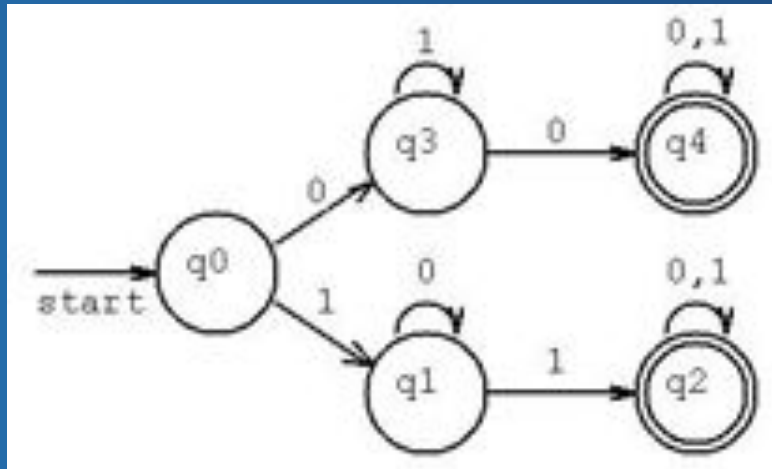
Multicast Channels



Publish Transport

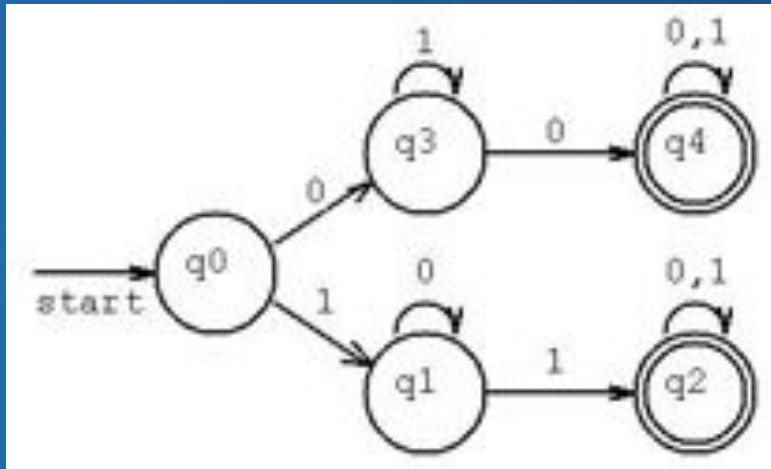
With **multicast channels** we can adjust subscriber discovery:

- Regex search for topic
- Get end states q2 & q4



Publish Transport

- Store channel in DHT
  - `e.g. DHT put (q2, my_channel) & (q4, my_channel)`
- Subscriber looks for channel
  - `e.g. DHT get (q2)`
- Actively join channel(s)



Problems

- We lose consistency
- Increased latency
- ~~Privacy issues~~
- No access control
  - Any peer can read any topic
  - Any peer can publish on any topic

Problems

- We lose consistency
- Increased latency
- ~~Privacy issues~~
- No access control
  - ~~Any peer can read any topic~~
  - Any peer can publish on any topic

Publish Access Control

- Publish on topic `/[pub_key]/nyt`
- Put signed channel entry into DHT
  - `(q2, (nyt_channel,sign))`
- Check validity
  - e.g. subscriber checks
  - e.g. dht checks

Problems

- We lose consistency
- Increased latency
- Privacy issues
- No access control
  - Any peer can read any topic
  - ~~Any peer can publish on any topic~~

Current Status

- Starting from existing Implementation
- Service & Client APIs are there ...
- … but do not work in all cases
- Only QoS Level 0 (best effort) available

Client API

- `GNUNET_MQTT_subscribe()`
- `GNUNET_MQTT_unsubscribe()`
- `GNUNET_MQTT_publish()`
- `GNUNET_MQTT_publish_cancel()`

Next Steps

1. Get tests running again
    a. 2 out of 5 still fail :-(
2. Add new features
    a. QoS level 1
    b. Retain flag
    c. Last will

Outlook

- Anonymous publisher discovery
- Publish transport (multicast)
- Publish access control (draft)

Further Reading

- MQTT Specification
- GNUnet-MQTT
  - https://github.com/vsaw/gnunet-mqtt
- Messaging Technologies for the Industrial Internet and the Internet of Things

So Long, and Thanks for All the Fish!

valentin.sawadski@mytum.de - david.froy@mytum.de

Partial Solution: Subscriber-side Blacklist

- MQTT blacklist in GNS with peer ids
- Blocks connection attempts