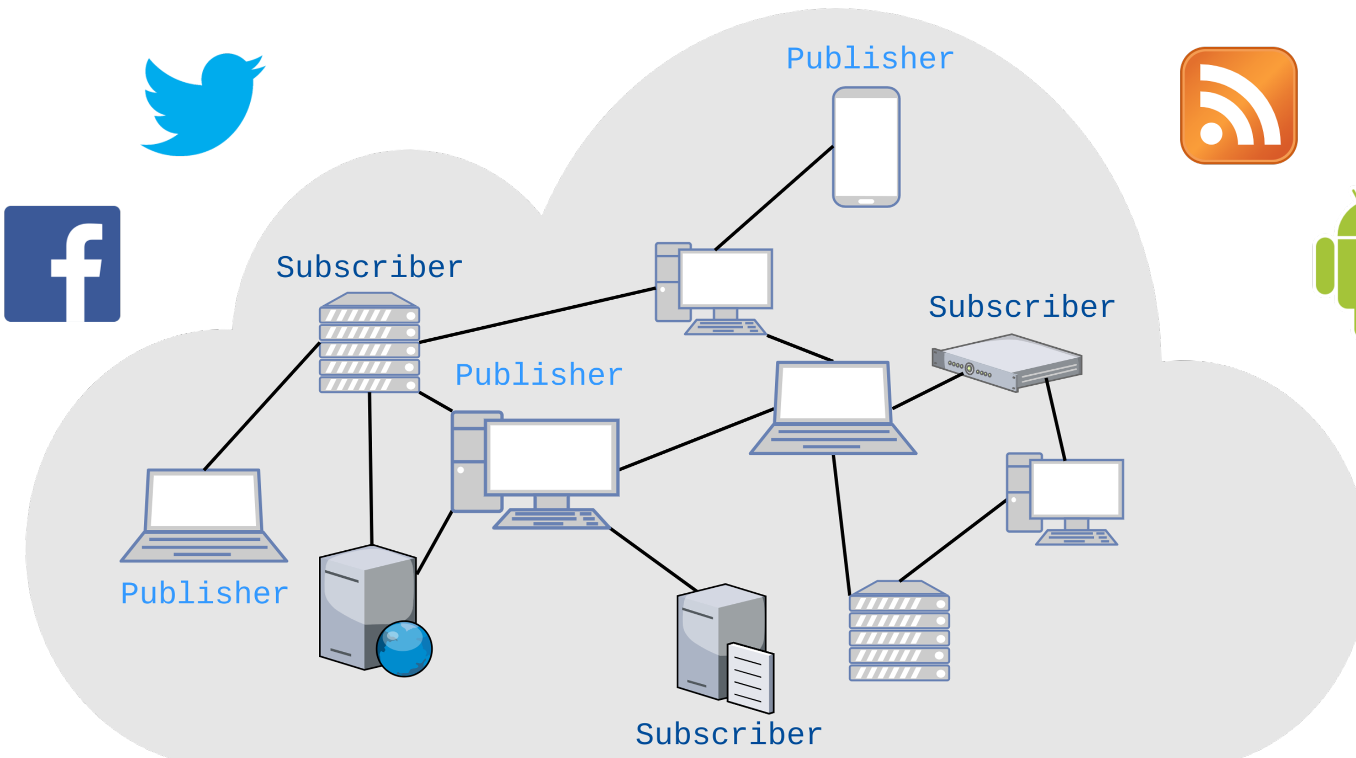
PulsarCast

Scaling Pub-Sub over the distributed web

João Gonçalo da Silva Antunes
Instituto Superior Técnico
INESC-ID

Motivation

World Wide Web



Pub-Sub Paradigm

- Communication paradigm providing **full decoupling** in:
  - Time
  - Space
  - Synchronisation

Problems

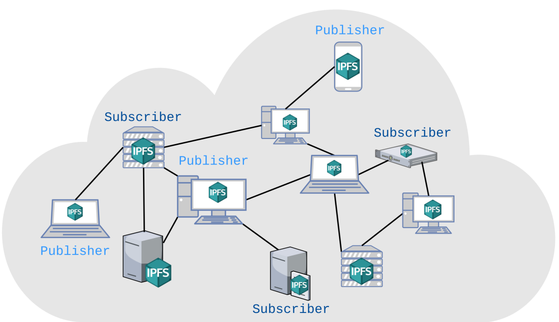
Lack of a pub-sub system that:

- Scales **for the web**
- Provides reliability:
  - Delivery guarantees
  - Data persistence

Objectives

- Decentralised pub-sub architecture using IPFS[1]:
  - Highly scalable
  - Reliable
  - Assures persistence
- Develop a system that meets these requirements
- Evaluate its performance on multiple environments

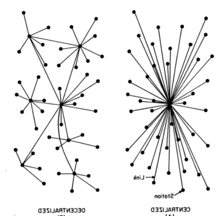
World Wide Web

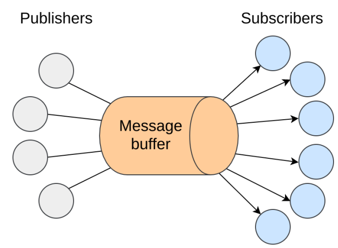


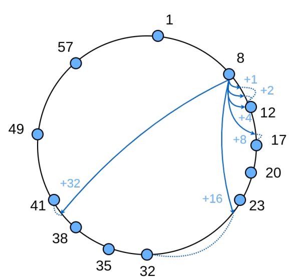
[1] - https://ipfs.io/

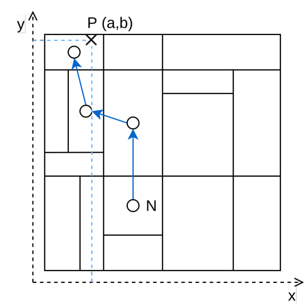
Related Work

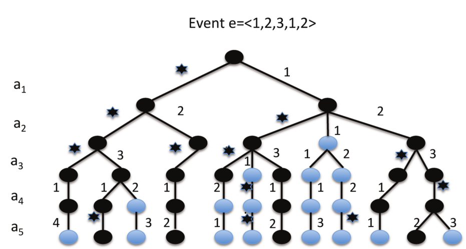
Pub-Sub Systems











Web Technologies

Design Dimensions

- Subscription model
- Network architecture
- Overlay structure
- Subscription management
- Event dissemination

Subscription Model

- Topic based
  - E.g. Redis[2], Scribe[1], Tera[7], Poldercast[10]
- Type based
  - E.g. Hermes[4]
- Content based
  - E.g. Gryphon[8], Siena[9], Meghdoot[2]

[2] - https://redis.io/

Topic Based

- ✔ A simple notion of group
- ✔ Members of the group receive every message
- ✔ Can build a complex hierarchy
- ✗ Lacks expressiveness

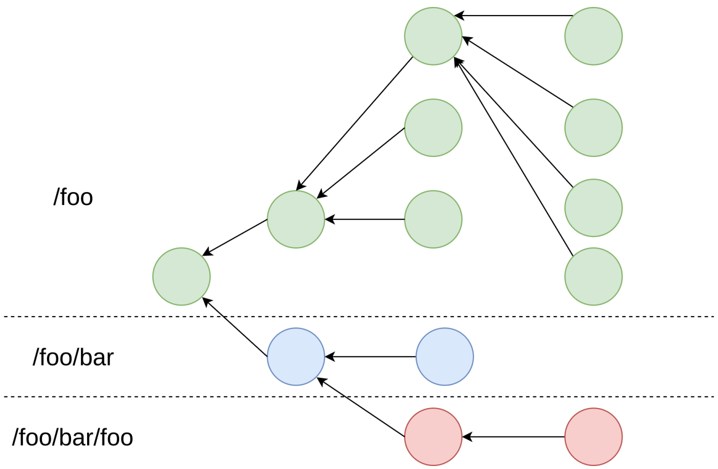


Fig.1: Events on a topic hierarchy

- **Note:** Type based can be seen as a special case of the topic based model fit for type based languages

TÉCNICO LISBOA

inesc id lisboa

Content Based

- ✔ Filter events based on their content
- ✔ Support for rich and flexible subscriptions
- ✗ Requires complex filtering and message forwarding

Message

```
{
  exchange: "Euronext Lisboa",
  company: "CTT",
  order: "buy",
  number: "100",
  price: "5.55",
}
```

Subscription

```
{
  exchange: "Euronext Lisboa",
  order: "buy",
  number: ">50",
  price: "<10",
}
```

Fig.2: JSON[3] subscription example

[3] - https://www.json.org/

Network Architecture

- Centralised:
  - E.g. Redis, Kafka[4], Gryphon
- Decentralised:
  - E.g. Scribe, Meghdoot, Poldercast

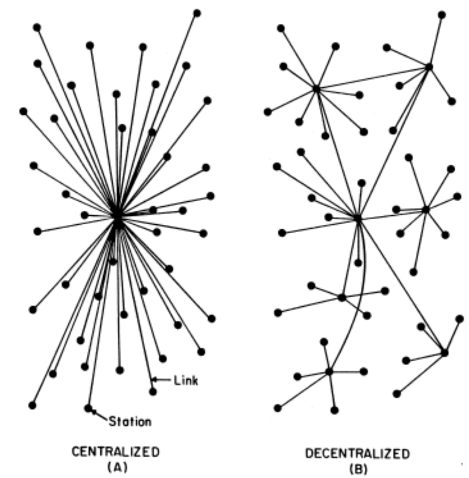


Fig.3: Network architecture overview

[4] - http://kafka.apache.org/documentation/#design

Centralised

- Focus:
  - ✔ Reliability (with replication)
  - ✔ Consistency
- Lacks:
  - ✗ Scalability
  - ✗ Data Throughput

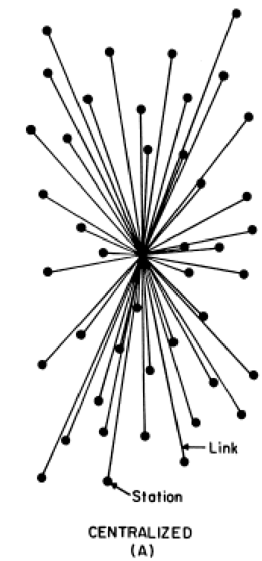


Fig.4: Centralised architecture example

Decentralised

- Focus:
  - ✔ Scalability
  - ✔ Data Throughput
- Lacks:
  - ✗ Reliability
  - ✗ Consistency
  - ✗ Persistence

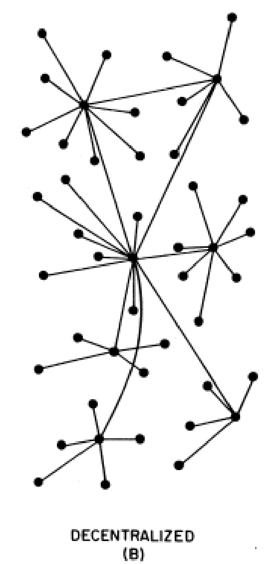
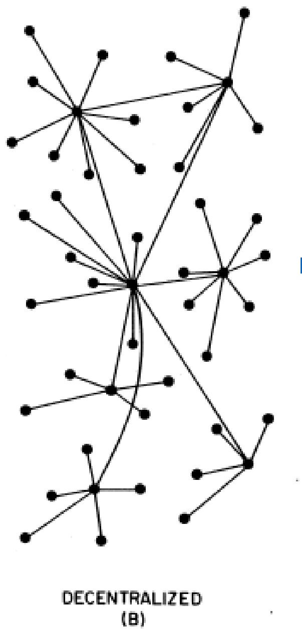
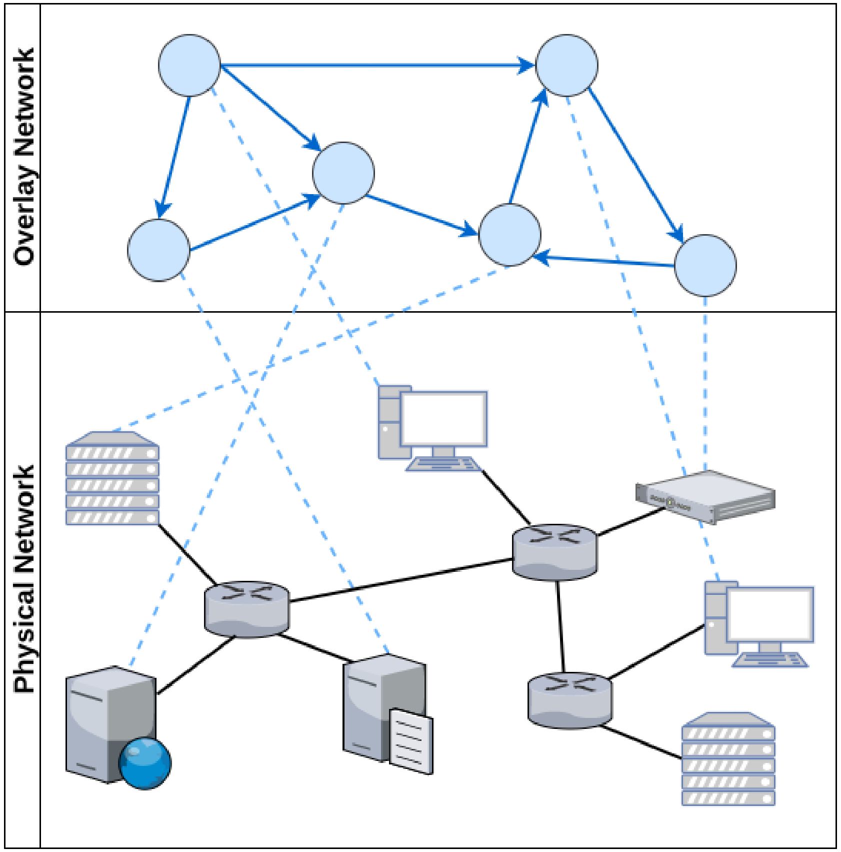


Fig.5: Decentralised architecture example

Network Overlay





Unstructured Overlay

- Each peer connects to a subset of nodes
- No clear structure or hierarchy
- Usage of gossip based membership protocols (e.g. Cyclon[11]) to help preserve:
  - Network diameter
  - Average degree

Structured Overlay

- Peers organise according to a specific structure (e.g. ring, tree, multi-dimensional space).
- Common approach is to use a Distributed Hash Table
  - E.g. Chord[5], Kademlia[3], CAN[6]

Structured Overlay - DHT

- Peer identifiers evenly spread across key space
- Ensures the content is evenly distributed
- Queries usually solved in logarithmic time

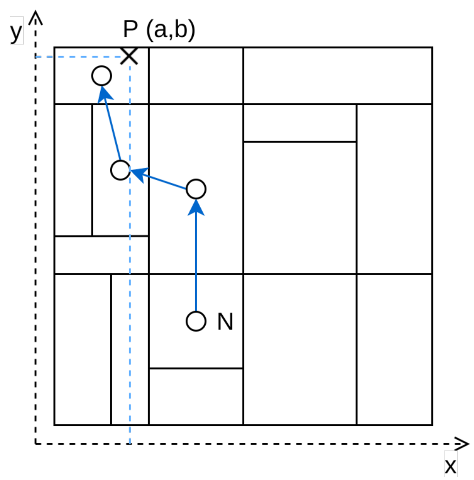


Fig.6: 2 dimensional CAN routing

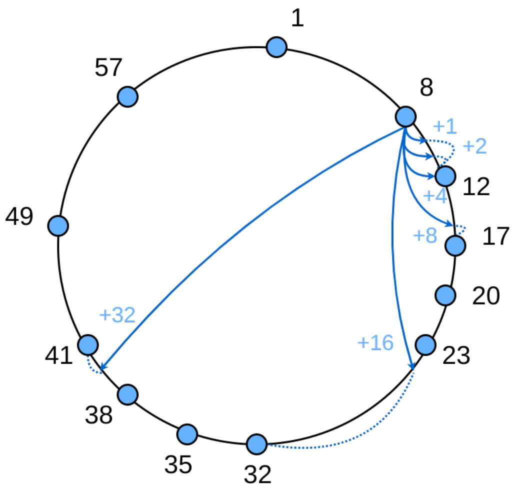


Fig.7: Chord ring

Subscription Management

Event Dissemination

Matching?

Distributed!

Ideally, evenly across the network

System Analysis

	Subscription Model	Architecture	Overlay Structure	Subscription Management	Event Dissemination
Gryphon	Content	(C) broker hierarchy	N/A	Each broker responsible for a subscription scheme	Tree hierarchy
Siena	Content	(C) broker mesh	N/A	Keep state at each node for subscription routes	Flood with cached state
Jedi	Content	(C) broker hierarchy	N/A	Keep state at each node for subscription routes	Tree hierarchy
Bayeux	Topic	Decentralised	Tapestry DHT	Rendezvous node	Multicast tree
Scribe	Topic	Decentralised	Pastry DHT	Rendezvous node	Multicast tree
Meghdoot	Content	Decentralised	CAN DHT	Point in CAN DHT	CAN routing
Hermes	Type	Decentralised	Pastry DHT	Rendezvous node	Multicast tree
Tera	Topic	Decentralised	Gossip based overlay	Unstructured overlay per topic	Random walks & flooding
Mercury	Content	Decentralised	Ring based DHTs	Overlay per attribute in schema	Route through ring overlays
Sub-2-Sub	Content	Decentralised	Ring based DHT & gossip overlay	Clustering of similar subscriptions	Gossip & ring overlay routing
Poldercast	Topic	Decentralised	Ring based DHT, Vicinity & Cyclon	Ring overlay per topic	Ring overlay routing

(C) – Centralised

System Analysis

	Relay Free Routing	Delivery Guarantees	Fault Tolerance
Gryphon	N/A	yes	Best effort
Siena	N/A	yes	Best effort
Jedi	N/A	yes	Best effort
Bayeux	No	yes	Best effort, no subscription persistence
Scribe	No	yes	Best effort, no subscription persistence
Meghdoot	No	yes	Replicated subscriptions
Hermes	No	yes	Best effort
Tera	No	no	Best effort
Mercury	No	yes	Best effort
Sub-2-Sub	No	no	Best effort
Poldercast	Yes	Yes (every publisher is a subscriber)	High resilience to churn, no subscription persistence

Delivery Guarantees – Event delivery guarantees under normal network conditions
Fault Tolerance – Mechanisms to guarantee successful event delivery and subscription matching under heavy churn

Web Technologies

- The Javascript[6] ecosystem
- New network protocols that facilitate P2P communication
- New P2P applications that leverage all of this
  - E.g. IPFS

[5] - https://www.ecma-international.org/publications/standards/Ecma-262.htm

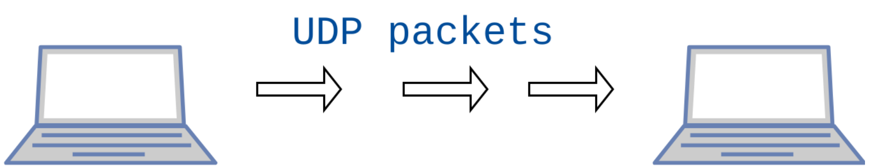
The Javascript Ecosystem

- **Javascript** is one of the main programming languages for the **web**[6]
- Thanks to **NodeJS**[7], now **runs in servers** also
- Its package manager, **NPM**[8], is one of the largest package registries in the world
- NPM powers a UNIX-like culture of small modules that work well together

[6] - https://insights.stackoverflow.com/survey/2017
[7] - https://nodejs.org
[8] - https://www.npmjs.com/

New Protocols

- **WebRTC**[9] made possible full-duplex communication between browsers without an intermediary.
- **QUIC**[10] and **uTP**[11] provided alternatives to TCP, bringing **reliability** and **order delivery** over **UDP**



[9] - https://www.w3.org/TR/webrtc/
[10] - https://datatracker.ietf.org/wg/quic/about/
[11] - http://www.bittorrent.org/beps/bep_0029.html

IPFS

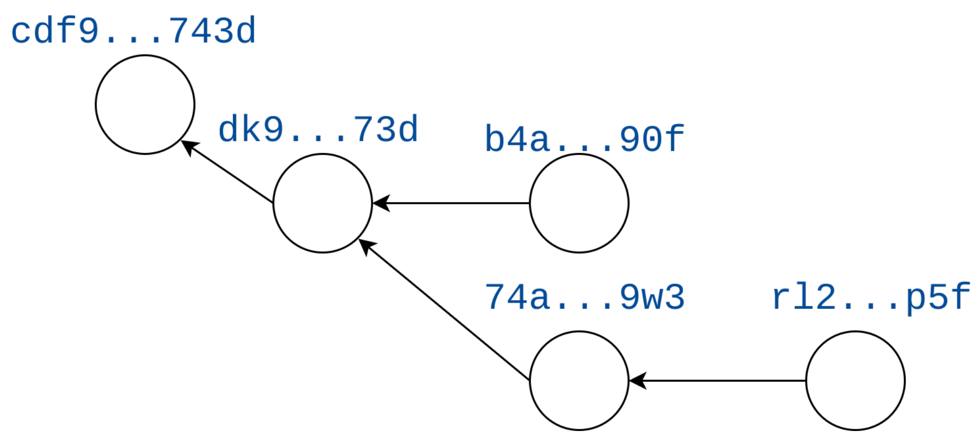
- A **P2P hypermedia** protocol designed to create a **persistent**, **content-addressable** network for the web
- Uses a modular approach through **libp2p**[12] to solve common challenges of P2P applications
- At its core, IPFS uses a **Merkle DAG**[13]

[12] - https://libp2p.io
[13] - https://github.com/ipld/specs/tree/master/ipld

Merkle DAG

- A graph structure used to store and represent **linked data**.
- Each node can be linked to based on the **hash of its content**.
- Referred to as **IPLD** in the IPFS ecosystem.
- Offers **immutability.**

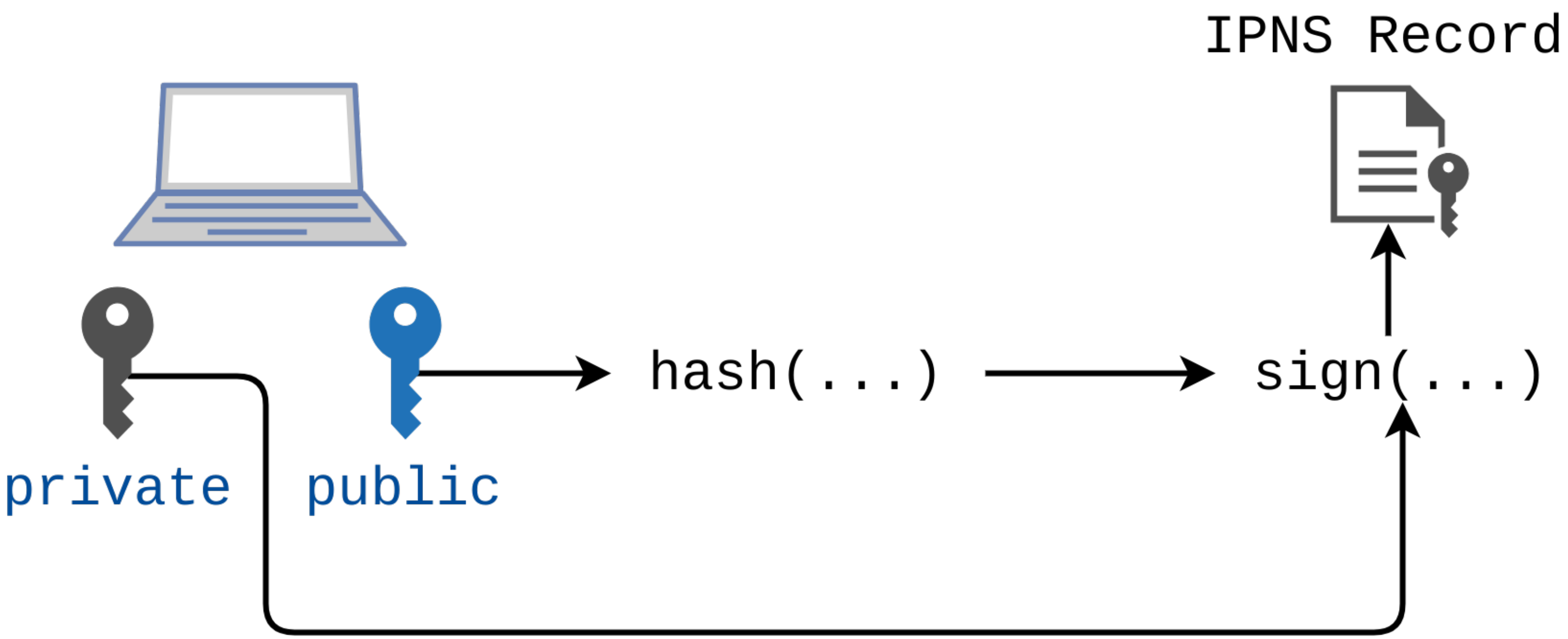
Merkle DAG




```json
{
  "content": {
    "key": "value"
  },
  "merkle-link": "cdf9...743d"
}
```

IPNS

- A way to offer **mutability**.



Proposed Solution

- A topic based, pub-sub module for the IPFS ecosystem that provides:
  - Data persistence
  - Delivery guarantees
  - High scalability

Architecture

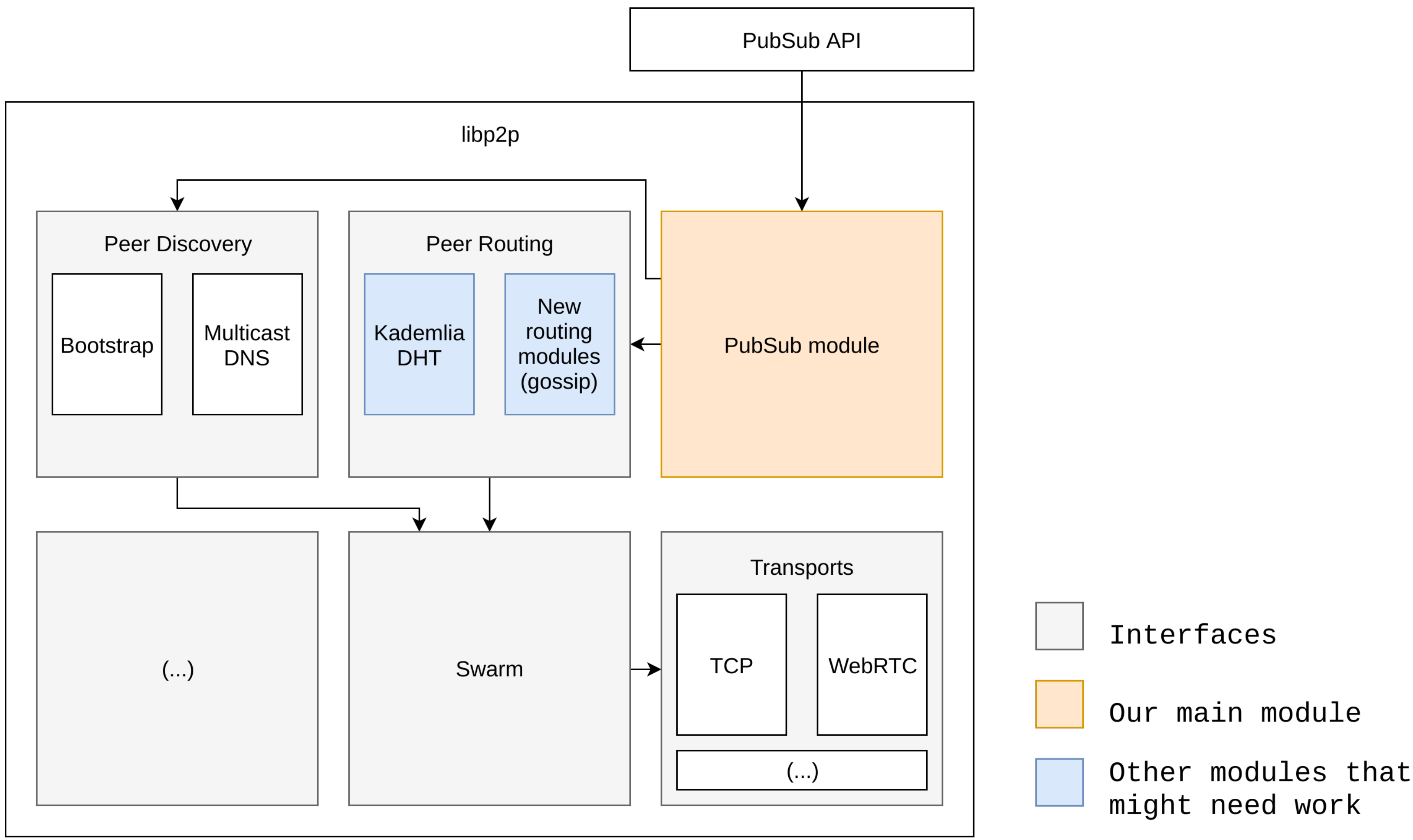


Fig.8: libp2p architecture

Libp2p Kademlia DHT

- put: insert a value with a given key.
- get: get a value of a given key.
- findPeer: find the peer with the given peer ID.
- **getClosestPeers:** find the *k* closest peers to a given key.
- **provide:** let the network know that this peer can also distribute a given key.
- **findProviders:** find providers for a given key.

Distributed Data Structures

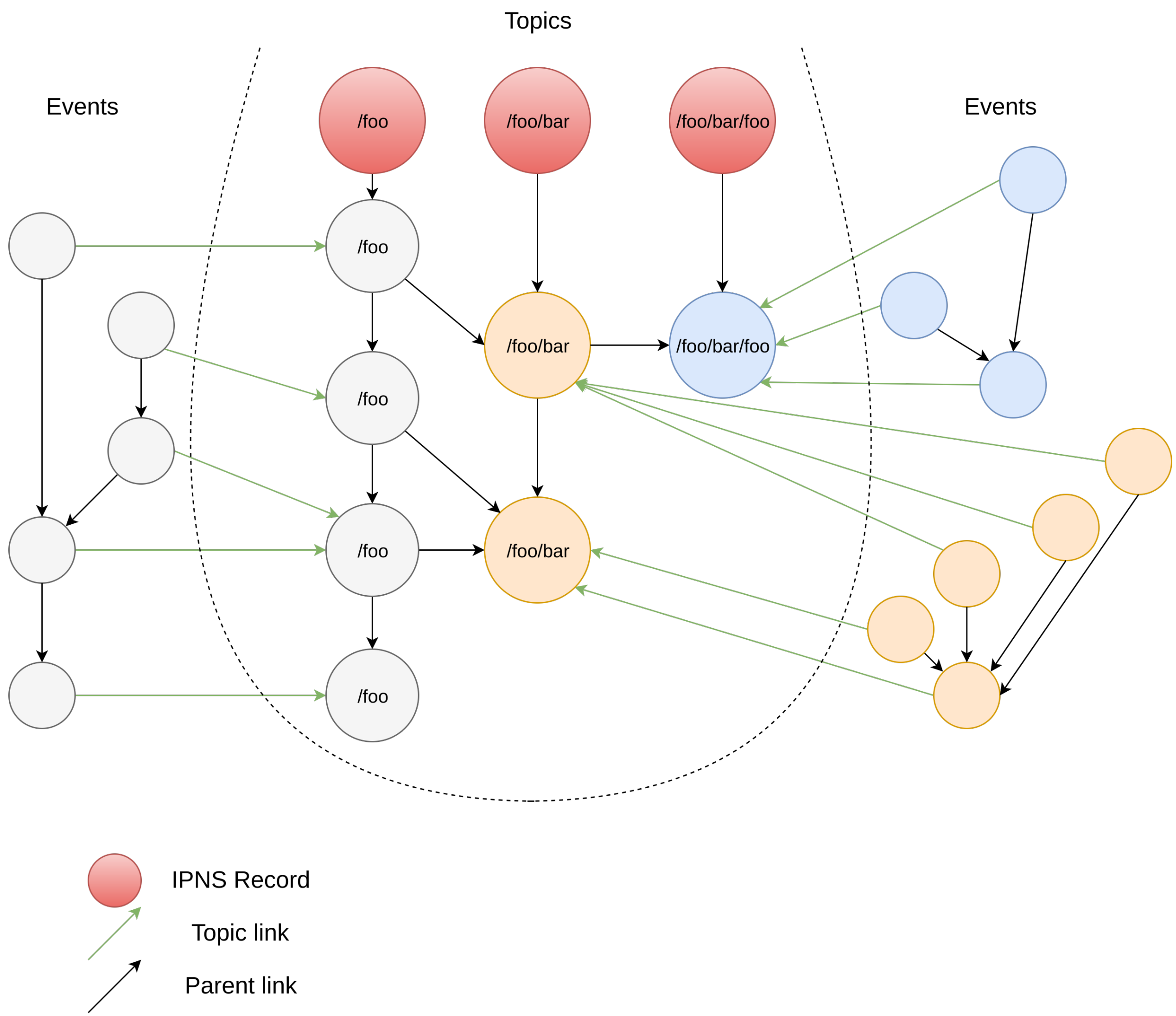
Topic Descriptor

```
{
  name: <topic-name>,
  metadata: <json-object>, // creation date, protocol version, etc.
  #: {
    <sub-topic-name>: <merkle-link to topic descriptor>,
    ...
  },
  parent: <merkle-link to topic descriptor>
}
```

Event Descriptor

```
{
  topic: {
    name: <topic-name>, // Name of the topic
    link: <merkle-link> // Link to the topic of this message
  },
  publisher: <publisher node ID>
  parent: <merkle-link to previous event>,
  metadata: <json-object>, // Timestamp and other relevant info
  payload: <json-object>, // The actual message content
}
```

Distributed Data Structures



Subscription Management

- Using the Kademlia DHT, build a tree structure when creating a new subscription.
-

Event Dissemination

- (mention matching)

Quality of Service

Evaluation

- Using either **PeerSim**[14] or **IPTB**[15]
- Using **synthetic datasets** of both **events** and **subscriptions**:
  - Of **different sizes**
  - With **different distributions**
- Using **our system**, IPFS **current pub-sub** and a **baseline**

[14] - http://peersim.sourceforge.net/
[15] - https://github.com/whyrusleeping/iptb

Evaluation - Metrics

- Ratio of **messages sent by each node** (correlated with the CPU, memory and bandwidth usages)**.**
- Ratio of **throughput speedup** vs **disk storage** used at each node.
- Ratio between **latency reduction** vs **disk storage** used at each node.
- Ratio of **subscriptions covered** under heavy **network churn**.
- Ratio of **subscriptions covered** after a severe **network partition** and its recovery.
- **State** of event streams at each node under heavy **network churn**.
- **State** of event streams at each node after a **network partition** and its recovery.

Conclusion

- Pub-sub systems are vital for distributed applications
- Lack of a system that:
  - Scales for the web
  - Focuses persistence and reliability
- Overview of relevant pub-sub systems and web technology
- Proposed solution of a decentralised pub-sub system for the web using IPFS

Thank you for your presence!

Questions?

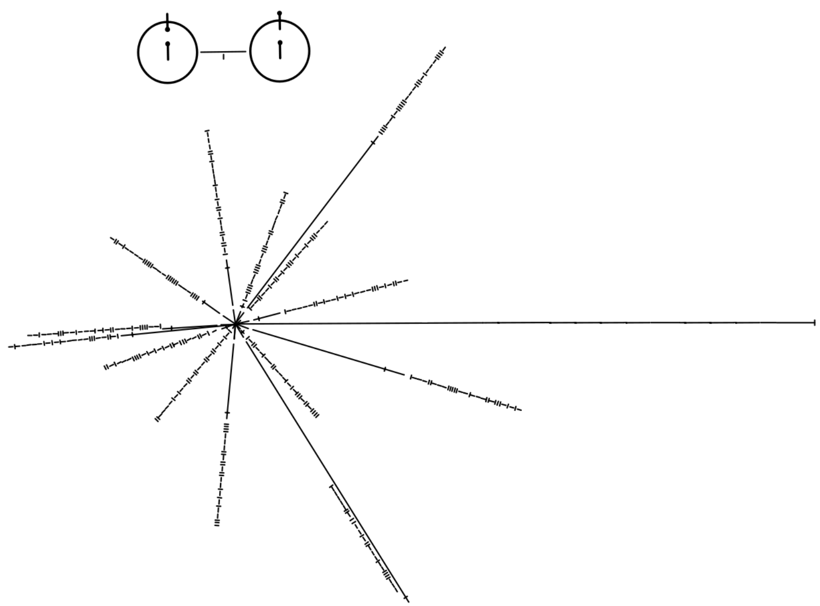


Fig.8: Pulsar map used for the pioneer plaque

References

1. - Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, and Antony Rowstron. Scribe:A large-scale and decentralized application-level multicast infrastructure. IEEE Journal on Selected Areas in Communication, 20, 2002.

2. - Abhishek Gupta, Ozgur D Sahin, Divyakant Agrawal, and Amr El Abbadi. Meghdoot: Content-Based Publish/Subscribe over P2P Networks. Springer LNCS, 3231/2004(Middleware 2004):254–273, 2004.

3. - Petar Maymounkov and David Mazières. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. Pages 53–65. 2002

4. - P. R. Pietzuch and J. M. Bacon. Hermes: A distributed event-based middleware architecture. Proceedings - International Conference on Distributed Computing Systems, 2002-Janua:611– 618, 2002.

5. - I Stoica, R Morris, D Karger, M F Kaashoek, and H Balakrishnan. Chord: A Scalable Peer-to-peer Pookup Service for Internet Applications. Sigcomm, pages 1–14, 2001.

6. - Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. ACM SIGCOMM Computer Communication Review, 31(4):161–172, 2001.

7. - R Baldoni, R Beraldi, V Q Ema, L Querzoni, and S Tucci-Piergiovanni. TERA: Topic-based Event Routing for peer-to-peer Architectures. 2007.

8. - Robert Strom, Guruduth Banavar, Tushar Chandra, Marc Kaplan, Kevan Miller, Bodhi Mukherjee, Daniel Sturman, and Michael Ward. Gryphon: An Information Flow Based Approach to Message Brokering. Arxiv preprint cs9810019, cs.DC/9810:1–2, 1998

9. - Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Design and evaluation of a wide-area event notification service. Foundations of Intrusion Tolerant Systems, OASIS 2003, 19(3):283–334, 2003.

10. - Vinay Setty and Maarten Van Steen. Poldercast: Fast, robust, and scalable architecture for P2P topic-based pub/sub. Proceedings of the 13th International Middleware Conference, pages 271–291, 2012.

11. - Spyros Voulgaris, Daniela Gavidia, and Maarten Van Steen. CYCLON: Inexpensive membership management for unstructured P2P overlays. Journal of Network and Systems Management, 13(2):197–216, 2005.

Topic Descriptor

```
{
  name: <topic-name>,
  metadata: <json-object>, // creation date, protocol version, etc.
  #: {
    <sub-topic-name>: <merkle-link to topic descriptor>,
    ...
  },
  parent: <merkle-link to topic descriptor>
}
```

Event Descriptor

```
{
  topic: {
    name: <topic-name>, // Name of the topic
    link: <merkle-link> // Link to the topic of this message
  },
  publisher: <publisher node ID>
  parent: <merkle-link to previous event>,
  metadata: <json-object>, // Timestamp and other relevant info
  payload: <json-object>, // The actual message content
}
```