Pulsarcast

Scalable and reliable pub-sub over P2P networks

João Gonçalo da Silva Antunes
Instituto Superior Técnico
INESC-ID

Motivation

Fig.1 - Illustration of how pub-sub powers many of the systems we rely everyday. [15] provides an example of such

Current Shortcomings

- **Centralised** pub-sub solutions commonly **lack**:
  - Scalability
  - Data throughput
- **Decentralised** pub-sub solutions commonly **lack**:
  - Reliability
  - Consistency
  - Delivery guarantees
  - Data persistence

Fig.2 - Centralised pub-sub diagram

Fig.3 - Centralised vs Decentralised architecture extracted from [24]

Our goals

Pulsarcast is a P2P pub-sub solution focused on:

- Eventual delivery guarantees
- Data persistence
- Scalability
- Leveraging network infrastructure and protocols already in place
- Reliability

Our contribution

- **Pulsarcast** [1] - the **specification** for a more efficient and reliable P2P pub-sub topic based system.
- **Distributed algorithms** and **data-structures** for subscription management and event dissemination based on IPFS' Merkle DAG.
- A **Javascript implementation module of Pulsarcast** [2] built on top of libp2p, easy to integrate with IPFS
- A **distributed test runner** [3] capable of running large scale test scenarios
- An easy to automate **test-suite** [4] based on a real world Reddit dataset

This work is partially described in a conference submission: João Antunes, Luís Veiga and David Dias. Pulsarcast - Scalable and reliable pub-sub over P2P networks, submitted to ACM/IFIP International Middleware Conference (MIddleware 2020)

All of our contributions are open source:
[1] - https://github.com/JGAntunes/pulsarcast
[2] - https://github.com/JGAntunes/js-pulsarcast
[3] - https://github.com/JGAntunes/ipfs-testbed, https://github.com/JGAntunes/helm-charts/tree/master/ipfs-testbed and https://github.com/JGAntunes/ipfs-testbed-cli
[4] - https://github.com/JGAntunes/pulsarcast-test-harness

Related Work

Pub-Sub Systems

Fig.4 - Overview of some of the figures used for our work and analysis of pub-sub systems

Web Technologies

Fig.5 - Overview of web technologies relevant for our work

Pub-Sub Systems - Design Dimensions

- Subscription model
  - Topic based, type based or content based
- Network architecture
  - E.g. centralised, broker based [4] [23], decentralised
- Overlay structure
  - Structured overlays (e.g. Kademlia DHT [11], CAN DHT [13], Chord [17] , Pastry [14]) or unstructured overlays (e.g. CYCLON [19] and [21])
- Subscription management
- Event dissemination

Fig.6 - Example of a simple Chord ring and the finger table of a node

Fig.7 - Example of a 2 dimensional CAN routing command

Pub-Sub Systems - Implementations

- Topic based - Clients subscribe to classes of events, usually identified by keywords
  - E.g. Scribe [5], Bayeux [22], TERA [2], SELECT [1], Poldercast [16], VCube-PS [7]
- Type based [9] - Brings the notion of a type scheme to a topic-based subscription model.
  - E.g. Hermes [12]
- Content based - Clients subscribe to events based on specific values (or ranges of values) on the properties of the events
  - E.g. Gryphon [18], Jedi [6], Meghdoot [10], Mercury [3], Sub-2-Sub [20], Siena [4]

Web Technologies

- The Javascript [5] ecosystem
  - NodeJS [6] brought even more ubiquity to Javascript
  - NPM [7] and its unique ecosystem
- New network protocols that facilitate P2P communication
  - E.g. WebRTC [8], uTP [9]
- New P2P applications and tools leveraging the above
  - E.g. IPFS [10], libp2p [11], BrowserCloud.js [8]

[5] - https://www.ecma-international.org/publications/standards/Ecma-262.htm
[6] - https://nodejs.org
[7] - https://www.npmjs.com/ and http://blog.npmjs.org/post/143451680695/how- many- npm- users- are- there
[8] - https://www.w3.org/TR/webrtc/
[9] - http://www.bittorrent.org/beps/bep_0029.html
[10] - https://ipfs.io
[11] - https://libp2p.io

Solution: Pulsarcast

Pulsarcast is a **peer to peer, pub-sub**, **topic-based** system focused on **reliability**, **eventual delivery** guarantees, and **data persistence**

Fig.8 - Pulsar “map” used to locate Earth and the Solar System in the Pioneer space probes launched in 1972 and 1973

Pulsarcast - Architecture - Network Overlays

Fig.9 - Representation of Pulsarcast's overlays

Pulsarcast - Distributed Data Structures

Event Descriptor

```
{
  publisher: <publisher-node-id>,
  author: <author-node-id>,
  parent: <merkle-link to previous event>,
  payload: <buffer>,
  topic: <merkle-link to topic>,
  metadata: <object> //Timestamp and other info
}
```

Fig.10 - JSON representation of Pulsarcast's event descriptor

Topic Descriptor

```
{
  name: <string>,
  author: <author-node-id>,
  parent: <merkle-link to previous version of topic>,
  #: {
    <sub-topic-name>: <merkle-link to sub topic>,
    ...
  },
  metadata: <object> //Timestamp and other info
}
```

Fig.11 - JSON representation of Pulsarcast's topic descriptor

Pulsarcast - Distributed Data Structures

Fig.12 - Representation of Pulsarcast's DAG

Pulsarcast - Subscription Management Algorithm

- Recursively, at each node:
  - Get the Topic Descriptor
  - If the node is not already part of the dissemination tree:
    - Locally, find the closest peer to the author
    - Send Join request

Fig.13 - Flow for querying a Topic/Event descriptor

Pulsarcast - Subscription Management Algorithm

Fig.14 - Overview of the flow for creating a new subscription

Pulsarcast - Event Dissemination Algorithm

Fig.15 - Event dissemination mechanism for a topic with order guarantee

Fig.16 - Event dissemination mechanism for a topic where everyone can publish

Implementation Details

Fig.17 - Our Pulsarcast module in the libp2p ecosystem

Implementation Details

Fig.18 - Condensed UML representation of the classes in Pulsarcast

Evaluation - Testbed

- Custom testbed using:
  - A version of **IPFS with Pulsarcast integrated**
  - **Docker** to create reproducible deployments
  - **Kubernetes** to orchestrate deployments
  - **Toxiproxy** to simulate adverse network conditions
  - ElasticSearch, Logstash, Kibana and Beats to collect, aggregate, visualise and analyse the results

Fig.19 - Overview of our ipfs-testbed deployment and our metrics/logs pipeline

Evaluation - Dataset

- As a dataset, we used a sample of Reddit comments from 2007.
  - 25000 comments (events)
  - 23 subreddits (topics)

Fig.20 - Subscription distribution per number of nodes

Fig.21 - Subscription distribution per topic

Fig.22 - Event distribution per topic in log scale

Evaluation - Test Runs

- Metrics to extract:
  - **Resource usage** for the whole cluster and per-node (average, P95 and P99):
    - CPU
    - Memory
    - Network
  - **QoS** (total, per topic and per node when applicable):
    - Events published per topic
    - Events received per topic
    - % of subscriptions fulfilled based on the events successfully published
    - % of subscriptions fulfilled based on the events injected in the system
    - Number of RPC messages sent
- **Test runs** (6 in total, with and without 500ms of network latency added):
  - Pulsarcast
  - Pulsarcast with order guarantee
  - Floodsub

Evaluation - Floodsub VS Pulsarcast

Fig.22 - Floodsub - Comparison of events fulfilled by topic in a log scale

Fig.23 - Pulsarcast - Comparison of events fulfilled by topic in a log scale

Evaluation - Floodsub VS Pulsarcast

Fig.24 - Floodsub - Comparison of percentage of events fulfilled by topic

Fig.25 - Pulsarcast - Comparison of percentage of events fulfilled by topic

Evaluation - Comparison Table

	Sub. Coverage	Sub. Coverage for Published Events	Effectively Published Events	Memory (GiB)*	Data Transmitted (MiB)*	CPU (vCPU)*
Floodsub	41%	41%	**100%**	21.15	6552.4	5.53
Pulsarcast	**99.99%**	**99.99%**	**100%**	31.92	2237.74	**3.77**
Pulsarcast with order G.	37%	82%	46%	**17.84**	**1050.4**	4.35
Floodsub with Latency	31%	31%	**100%**	**15.95**	6474.79	4.74
Pulsarcast with Latency	**52%**	**86%**	60%	35.8	3423	**1.26**
Pulsarcast with order G. and Latency	32%	62%	53%	19.99	**1595.6**	4.33

* max values for the total system (sum of all the resources per node)

Conclusion

- **Pub-sub** systems are **vital** for distributed applications
- **Pulsarcast** seeks to bring key aspects of centralised solutions to the decentralised world:
  - **Persistence**
  - **Reliability**
  - **Eventual delivery guarantees**
- Our implementation on top of libp2p not only **performs better than the native libp2p floodsub** solution, it provides a better QoS.

Q&A

References

1. Nuno Apolonia, Stefanos Antaris, Sarunas Girdzijauskas, George Pal- lis, and Marios Dikaiakos. SELECT: A distributed publish/subscribe notification system for online social networks. Proceedings - 2018 IEEE 32nd International Parallel and Distributed Processing Symposium, IPDPS 2018, pages 970–979, 2018.
2. R Baldoni, R Beraldi, V Q Ema, L Querzoni, and S Tucci-Piergiovanni. TERA: Topic-based Event Routing for peer-to-peer Architectures. 2007.
3. Ar Bharambe, Sanjay Rao, and Srinivasan Seshan. Mercury: a scalable publish-subscribe system for internet games. 1st Workshop on Network and Systems Support for Games (NetGames '02), pages 3–9, 2002.
4. 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.
5. 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.
6. G. Cugola, E. Di Nitto, and A. Fuggetta. The JEDI event-based infrastructure and its application to the development of the OPSS WFMS. IEEE Transactions on Software Engineering, 27(9):827–850, 2001.
7. João Paulo De Araujo, Luciana Arantes, Elias P. Duarte, Luiz A. Rodrigues, and Pierre Sens. A Publish/Subscribe System Using Causal Broadcast over Dynamically Built Spanning Trees. Proceedings - 29th International Symposium on Computer Architecture and High Performance Computing, SBAC-PAD 2017, pages 161–168, 2017.
8. David Dias and Luís Veiga. BrowserCloud.js: A distributed computing fabric powered by a P2P overlay network on top of the web platform. Proceedings of the ACM Symposium on Applied Computing, pages 2175– 2184, 2018.
9. Patrick Eugster, Rachid Guerraoui, Joe Sventek, and Agilent Laboratories Scotland. Type-Based Publish/Subscribe. Technical report, Swiss Federal Institute of Technology, Lausanne, 2000.
10. 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.
11. Petar Maymounkov and David Mazières. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. pages 53–65. 2002.
12. 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.
13. 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.
14. Antony Rowstron and Peter Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. Number November 2001, pages 329–350. 2001.

References

15. Vinay Setty, Gunnar Kreitz, Roman Vitenberg, Maarten van Steen, Guido Urdaneta, and Staffan Gimåker. The hidden pub/sub of spotify. In Proceedings of the 7th ACM international conference on Distributed event-based systems - DEBS '13, page 231, New York, New York, USA, 2013. ACM Press.
16. Vinay Setty and Maarten Van Steen. Poldercast: Fast, robust, and scalable architecture for P2P topic-based pub/sub. Proceedings of the 13th . . . , pages 271–291, 2012.
17. I Stoica, R Morris, D Karger, M F Kaashoek, and H Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. Sigcomm, pages 1–14, 2001.
18. 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.
19. 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.
20. Spyros Voulgaris, Etienne Rivière, Anne-Marie Kermarrec, and Maarten Van Steen. Sub-2-Sub: Self-Organizing Content-Based Publish and Subscribe for Dynamic and Large Scale Collaborative Networks. Technical report, 2005.
21. Huanyang Zheng and Jie Wu. NSFA: Nested Scale-Free Architecture for scalable publish/subscribe over P2P networks. Proceedings - International Conference on Network Protocols, ICNP, 2016-Decem:1–10, 2016.
22. Shelley Q Zhuang, Ben Y Zhao, Anthony D Joseph, Randy H Katz, and John D Kubiatowicz. Bayeux. In Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video - NOSSDAV '01, number June, pages 11–20, New York, New York, USA, 2001. ACM Press.
23. Nejc Zupan, Kaiwen Zhang, and Hans Arno Jacobsen. Demo: Hyper- PubSub: a decentralized, permissioned, publish/subscribe service using blockchains. Middleware 2017 - Proceedings of the 2017 Middleware Posters and Demos 2017: Proceedings of the Posters and Demos Session of the 18th International Middleware Conference, pages 15–16, 2017.
24. P. Baran, "On Distributed Communications Networks," 1964.

Appendix A - Merkle DAG

Fig.26 - Graph visualisation of a Merkle DAG and its respective hash function dependencies

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

Fig.27 - JSON representation of a Merkle node with a Merkle link

Appendix B - Detailed Topic Descriptor

```
{
  "name": <string>,
  "author": <peer-id>,
  "parent": {                        //The parent link for this topic
    "/": <topic-id>
  },
  "#": {                             //Sub topic links
    "meta": {                          //Meta topic
      "/": "zdpuAkx9dPaPve3H9ezrtSipCSUhBCGt53EENDv8PrfZNmRnk"
    },
    <topic-name>: {
      "/": <topic-id>
    },
    ...
  },
  "metadata": {
    "created": <date-iso-8601>,
    "protocolVersion": <string>,  //Pulsarcast protocol version
    "allowedPublishers": {        //If enabled, whitelist of allowed publishers
      "enabled": <boolean>,
      "peers": [ <peer-id> ]
    },
    "requestToPublish": {         //Enable request to publish
      "enabled": <boolean>,
      "peers": [ <peer-id> ]      //Optional whitelist able to request
    },
    "eventLinking": <string>,     //One of: LAST_SEEN, CUSTOM
  }
}
```

Fig.28 - Topic descriptor schema in a JSON based format

Appendix C - Detailed Event Descriptor

```
{
  "name": <string>,
  "publisher": <peer-id>,          //Peer who published the event
  "author": <peer-id>,             //Author of the event
  "parent": {                      //The parent link for this event
    "/": <topic-id>
  },
  "topic": {
    "/": <topic-id>
  },
  "payload": <binary-data>
  "metadata": {
    "created": <date-iso-8601>,
    "protocolVersion": <string>,  //Pulsarcast protocol version
  }
}
```

Fig.29 - Event descriptor schema in a JSON based format

Appendix D - Join Request Algorithm

```
Function ReceivedJoin(fromNodeId, topicId)
    Data: nodeId = node id of this node
    Input: topicId = topic id
    Input: fromNodeId = node who we got the join request from
    begin
        topicData ← GetTopicData(topicId);
        if fromNodeId ≠ nodeId then
            AddToChildren(t, fromNodeId);          // Add as child in dissemination tree
            if topicData.author == nodeId then ;    // This node is author of the topic

                return
            end
            if GetParents(topicId) ≠ null then ;    // Already part of dissemination tree

                return
            end
        else
            if topicData.author == nodeId then ;    // This node is author of the topic

                return
            end
        end
        peer ← GetClosestKnownPeer(topicData.author);
        AddToParents(topicData.id, peer);          // Add as parent in dissemination tree
        SendRPC(topicData.id, peer);
    end
```

Fig.30 - Join request handler for each node

Appendix E - Event Handling Algorithms

```
Function ReceivedEvent(fromNodeId, eventData)
    Data: nodeId = node id of this node
    Input: fromNodeId = node who we got the event from
    Input: eventData = event descriptor
    begin
        topicData ← GetTopicData(eventData.topicId);
        if AllowedToPublish(nodeId, topicData) then
            SendEvent(fromNodeId, eventData);
        else
            if AllowedToRequestToPublish(nodeId, topicData then
                SendRequestToPublish(eventData);   // Send request to publish to parent
                 node

            end
        end
    end
```

Fig.31 - Event handler for each node

```
Function SendEvent(eventData)
    Data: nodeId = node id of this node
    Input: fromNodeId = node who we got the event from
    Input: eventData = event descriptor
    begin
        topicData ← GetTopicData(eventData.topicId);
        if IsNewEvent(eventData) then
            linkedEvent ← LinkEvent(eventData);                    // Add parent link
            StoreInDHT(linkedEvent);
        end
        if (IsSubscribed(eventData.topicId) == true) then
            EmitEvent(eventData.topicId, eventData);
        end
        for peer ← GetChildren(eventData.topicId), GetParents(eventData.topicId) do
            if fromNodeId ≠ peer then ;                    // Do not send the event back

                SendRPC(eventData, peer);
            end
        end
    end
```

Fig.32 - Event forwarding function

Appendix F - Pulsarcast RPC/event comparison

Fig.33 - Comparison of events received and RPC messages injected in the system

Appendix G - Pulsarcast VS Floodsub w/ latency

Fig.34 - Floodsub with latency - Comparison of events fulfilled by topic in a log scale

Fig.35 - Pulsarcast with latency - Comparison of events fulfilled by topic in a log scale

Pulsarcast

Fig.36 - Floodsub with latency - Comparison of percentage of events fulfilled by topic

Fig.37 - Pulsarcast with latency - Comparison of percentage of events fulfilled by topic

Appendix H - Pulsarcast w/ order guarantee

Fig.38 - Pulsarcast w/ order guarantee - Comparison of events fulfilled by topic in a log scale

Fig.39 - Pulsarcast w/ order guarantee and lat. - Comparison of events fulfilled by topic in a log scale

Fig.40 - Pulsarcast w/ order guarantee - Comparison of percentage of events fulfilled by topic

Fig.41 - Pulsarcast w/ order guarantee and lat. - Comparison of percentage of events fulfilled by topic