UPSYCLE: Ubiquitous Publish-Subscribe Infrastructure for Collaboration on Edge Networks

TG x Thoth

2020

Table of Contents

1 Abstract

The UPSYCLE protocol suite enables decentralized and asynchronous topic-based publish-subscribe communication across the internet and on edge networks. To achieve this, it combines trust-aware peer sampling, privacy-preserving subscription clustering, and reliable causal delivery in a two-tier P2P system that consist of a core network of always-on nodes and edge networks of mobile nodes where core nodes provide store-and-forward proxy services to mobile nodes to ensure reliable asynchronous communication across the internet.

2 Introduction & related work

Publish-subscribe [10] is a widely used communication paradigm over the internet where *publishers* publish events, while *subscribers* subscribe to these and receive *event notifications*. Various approaches have been proposed: *topic-based*, *content-based*, and *type-based*. In the topic-based model topics are analogous to communication channels where *subscribers* subscribe to *topics* of interest, and receive *event notifications* from *publishers* for their subscribed topics,

Topic-based publish-subscribe has wide range of applications including event notifications, database updates, messaging, social interactions and group collaboration, and more recently in Internet of Things and Distributed Ledger Technologies as well [24].

To address the scalability and reliability issues of centralized approaches, publish-subscribe in a decentralized setting has been extensively researched, however often times they lack attack resiliency and privacy properties that would be necessary for real-world deployments.

2.1 Desirable properties

Decentralized publish-subscribe overlays should satisfy a number of desirable properties:

- *scalability* in terms of number of nodes, topics, subscribers per topic, and subscriptions per node
- *relay-free routing* (also known as *topic-connectivity*, or *topic-connected overlay* (TCO), *traffic confinement*, and *noiselessness*): only subscribers of a topic participate in routing events for that topic
- *bounded node degrees*
- *completeness* of event dissemination: all published events should be delivered to all subscribers
- *low duplication factor* of event dissemination

- *low latency* of event dissemination
- *robustness* of the overlay: quick recovery after failures and churn

Equally important but not often considered:

- *attack resilience*: the overlay should be resilient to known attacks
- *subscription privacy*: nodes' subscriptions should be private, only common topic subscriptions between two nodes should be discoverable

Many of these properties are at odds with each other, and thus balancing these trade-offs is a key task of publish-subscribe designs. A number of different designs have been proposed that pick different trade-offs depending on their design goals.

2.2 Topic-connected overlays

Decentralized approaches are based either on structured or unstructured overlay networks. Structured approaches are scalable with regards to node degrees, however peers have to *relay* traffic they are not interested in and thereby have increased event delivery latency and traffic overhead. Unstructured approaches are able to achieve or approximate a topic-connected overlay [6, 8, 21, 25], however keeping node degrees bounded is a challenge in such overlays.

To achieve topic-connectivity, [6] suggests the following steps:

- *Interest clustering* to cluster peers with common topic subscriptions
- *Inner-cluster dissemination* to disseminate events inside clusters
- *Outer-cluster routing* to route to members of a cluster

2.3 Interest clustering

Interest clustering reduces node degrees by connecting to nodes with overlapping subscriptions that can deliver events from multiple topics over a single connection.

Gossip-based approaches such as [14, 25] typically achieve this by combining a gossip-based *random peer sampling* (RPS) protocol [15] with a gossip-based clustering protocol.

In gossip-based peer sampling protocols each node maintains a partial view of other nodes participating in the network, and exchanges this with a random node at each gossip round. However, the random selection process can be trivially biased by malicious peers who can launch an *eclipse attack* or *hub attack* to isolate nodes [16]. Various approaches have been proposed to tackle this issue, e.g. based on *social network analysis* [16], *stream samplers* [2, 7], and *social trust* [12].

Another important concern with clustering protocols is ensuring subscription privacy. The naive approach of exchanging full subscription sets [25] has both privacy and scalability issues. This can be improved upon by exchanging Bloom filters of subscriptions instead [21], which can be further enhanced by randomization to achieve differential privacy [1]. The loss of accuracy can be compensated by a *private set intersection* protocol to determine the exact set of overlapping topics upon establishing a connection to a peer. The Bloom filter-based protocol proposed by [20] achieves this in an efficient manner.

2.4 Overlay structure

TCOs are organized as separate suboverlays per topic where interest clustering helps with reducing node degrees by reusing links between nodes with overlapping subscriptions to deliver events for multiple topics [6, 18, 21, 25].

Using many suboverlays can quickly lead to high cumulative maintenance costs, therefore in order to achieve scalability in terms of number of subscriptions per node, it's important to reduce per-topic maintenance costs as much as possible by employing simple yet robust dissemination structures, such as per-topic rings with random shortcuts [25] .

2.5 Routing

Outer-cluster routing is necessary to find members for topics in order to join their suboverlay.

The interest clustering protocol can take care of this, which can be expedited by making use of topic priorities in the protocol to be able to prioritize finding members for a newly subscribed topic. However, this approach does not work well when subscriptions sets are represented as Bloom filters.

Another approach is creating a Small-World Interest-Close Overlay (SWICO) [5, 8] where nodes maintain both short-distance connections to interest-close nodes, and long-distance connections to nodes with dissimilar interests, thereby creating a small-world network with low diameter and fast routing.

2.6 Event dissemination

The *inner-cluster dissemination* protocol should deliver all events to all subscribers (a.k.a. *completeness*, or *totality*) while keeping the event duplication factor low.

Subscribers should be able to detect missed events and re-request them from other subscribers. By embedding causality information in events, a reliable causal delivery protocol can establish partial ordering of events and detect and re-request missed ones [17], while a probabilistic reliable broadcast protocol can ensure totality, consistency, and validity properties [13].

2.7 Group encryption

Desirable security properties for decentralized group communication include: authentication, confidentiality, integrity, eventual consistency, forward secrecy (FS) and post-compromise security (PCS), as well as dynamic group membership.

Many existing protocols aim to solve part of these problems, but either relies on a centralized entity or lacks certain security properties.

The Signal group protocol relies on two-party communication channels, and thus does not scale to large groups, the Sender Keys protocol does not provide PCS, Megolm lacks forward secrecy, while Message Layer Security (MLS) relies on a central entity for total ordering [26].

The Decentralized Secure Group Messaging protocol proposed in [26] aims to address these issues.

3 Design

3.1 Design requirements

The design requirements for UPSYCLE and how we achieve them are the following:

Scalability is achieved by minimizing overlay & suboverlay maintenance and by efficient dissemination in suboverlays

Relay-free routing is enforced by creating a suboverlay for each topic

Bounded node degrees are achieved via interest clustering

Low latency & duplication factor as much as the scalability constrains of suboverlay maintenance allows

Reliable delivery & causal order is ensured by *causal barriers* and a *reactive error recovery* mechanism

Subscription privacy subscriptions should be private and only common group membership between peers should be able to be discovered

Resiliency the use of explicit trust networks make the protocols more resilient to attacks

Minimalism we strive to minimize protocol complexity and hardware resources, e.g. by avoiding expensive Proof-of-Work computations and by employing a two-tier network to minimize resource requirements for mobile nodes

Offline-first nodes on edge networks should have a copy of all the data they subscribed to and should be able to communicate directly and opportunistically synchronize with the core network

3.2 Design overview

UPSYCLE is a decentralized publish-subscribe system designed with the requirements of resource constrained and intermittently connected mobile devices in mind. Since mobile devices are bandwidth and battery constrained, we propose a two-tier P2P system, where a P2P *core network* runs a set of P2P protocols, while mobile devices form *edge networks* for local interaction and connect to one or more remote *proxies*, which are always-on nodes that participate in the core P2P network and act as store-and-forward proxies for mobile nodes. This way it's sufficient for a mobile node to establish a single connection to a proxy to reach remote nodes.

It's important to note here that these proxies only perform store-and-forward message relaying, and cryptographic user and group identities are independent of them. This allows a mobile node to choose a different proxy at any moment, or even to use multiple proxies for redundancy.

This approach avoids the issues of centralized and federated systems (such as Facebook and Matrix) where user data and identities are tied to a specific server provider, and thus migration to a different provider is either difficult or impossible. In addition, the use of proxies also provides location privacy to users, i.e. a user's IP address is never revealed, except to the user's own proxy, which results in VPN-like privacy protections.

Next to connecting to proxies, mobile nodes can also maintain direct P2P connections with other nodes on the local network where they participate in similar P2P protocols to the ones in the core network. This allows local collaboration, even without internet connectivity.

There can be serious privacy implications of exposing group membership, especially on local networks. Therefore, group discovery, both in core and edge networks, is based on a Private Set Intersection (PSI) protocol. In groups where pseudonymity is desired, even the discovery of other group members on local networks could be problematic, and thus users should be able to opt in to local group discovery on a per-group and per-network basis.

3.3 P2P transport

Peers in the network establish end-to-end encrypted P2P connections among each other. Gossip-based peer sampling and dissemination protocols rely on these links to reach other peers in the network. Since gossip-based protocols need to establish new connections frequently to other peers, it's important to minimize the connection setup overhead

[11, 19], which includes a TCP handshake, a Diffie-Hellman key exchange, and negotiation of cryptographic parameters. Using UDP instead of TCP, as well as protocols with optimized cryptographic handshakes, caching encryption keys for session resumption, and keeping connections open for reuse are techniques that help to reduce the connection setup overhead.

TLS and DTLS are two commonly used transport security protocols for TCP and UDP, respectively. The recently introduced version 1.3 of TLS brings many improvements to the handshake process, reducing it to 1-RTT for new connections and 0-RTT for connection resumption. Version 1.3 of DTLS makes similar improvements for TLS over UDP.

Wireguard [9] is a UDP-based encrypted tunnel protocol based on the Noise Protocol Framework [22]. It is a considerably simpler protocol than DTLS, with security improvements and fast, 1-RTT handshakes. However, it requires setting up static tunnels among a fixed set of hosts, and thus it is not suitable for a P2P setting where the network is dynamic and the nodes are not all known before.

For these reasons initially we rely on TLS 1.3 and later DTLS 1.3 once it becomes available.

3.4 Interest clustering

As in [25], interest clustering is based on a combination of two gossip protocols, random peer sampling [15] and a similarity-based clustering protocol.

In order to make the peer sampling protocol resilient to attacks [2, 7, 16], we employ a stream sampler as specified in [2], which filters out over-represented nodes from a stream of incoming node IDs. The peer sampling protocol also need to limit push from other peers to limit the influence of any one peer. In contrast to [7] which achieves this by using proof of work, we opt for pull-only gossip in order to reduce the computational requirements of the protocol.

The clustering protocol uses Bloom filters to represent subscriptions of a node, as in [21], to make the exchange of subscription information scalable & privacy-protecting. To provide subscription privacy with differential privacy guarantees, we randomize the Bloom filters with random bit flips as described in [1]. The clustering protocol then computes subscription similarity based on the similarity between randomized Bloom filters.

Furthermore, we employ an explicit trust network to bias peer selection both in the peer sampling in clustering protocols, as suggested by [12]. In contrast to [12], we use asymmetric trust values between peers, and omit transmitting trusted paths in the protocol in order to avoid issues regarding exposing trust relationships and values between peers, to make the protocol resilient to malicious nodes trying to spread false information, and to make the protocol simpler.

3.5 Routing

In order to route join requests to members of the target topic suboverlay, we need an efficient routing mechanism. Small-world networks have low diameter and provide fast routing and thus would be a desirable structure for the overlay.

To achieve a small-world network topology, as part of the clustering protocol each node maintains a set of *fingers* (nodes with the most dissimilar interest) that serve as long-distance routing links, in addition to the most similar nodes that provide short-distance routing to nodes with overlapping interest, in a similar fashion to [5]. This prevents the overlay from forming *weak bridges* (small number of connections between clusters) and keeps the overlay diameter low.

3.6 Event dissemination

For event dissemination in suboverlays, UPSYCLE uses a combination of two approaches: deterministic dissemination over a ring with random shortcuts, as described in [25]. This approach is simple and comes with minimal maintenance overhead, while being reasonably efficient in terms of latency and duplication factor.

By minimizing suboverlay maintenance overhead, the system can scale with the number of subscriptions per node, at the expense of being less efficient in terms of latency and duplication factor.

3.7 Reliable causal delivery

In order to ensure completeness of dissemination and causal ordering of events, UPSYCLE uses a handful of approaches.

We use *causal barriers* [3, 4, 23] to ensure causal ordering of events in a topic: each event includes its direct dependencies that must be delivered before. If any dependency of an event is not yet received by a node, it needs to explicitly request those from other subscribers of the topic before it can deliver the event. Since event delivery can be delayed due to the different paths events can take, requesting missing dependencies should be only done after a delay, as part of a *reactive error recovery* mechanism described in [17].

In practice, this means that each event has an ID based on its content hash, and the following header fields that facilitate causal ordering and allow detecting missed messages:

Direct dependencies List of event IDs that are direct dependencies of this event. This ensures causal delivery.

Concurrent events List of event IDs that are independent but concurrent to this event. This allows nodes to detect missed events unrelated to the current one, and also serves as an implicit acknowledgement of the receipt of the referenced events.

Explicit acknowledgements can also be used to ensure event delivery, these are empty messages that list the events to be acknowledged as their direct dependencies.

3.8 Event synchronization

Synchronization of received events among two peers is necessary in a couple of scenarios. A new subscriber who has just subscribed to a topic may want to receive past events sent to the group. Similarly, rejoining subscribers would want to synchronize events they missed. Furthermore, during normal operation of the dissemination protocol, it might happen that an event is not delivered to a subscriber, which can be detected since causal dependency information is included in each event. In this case one can request missed events from other subscribers of the topic.

3.9 Group encryption & membership

We use the decentralized secure group messaging protocol suite described in [26].

The main components of this protocol suite are:

Authenticated Causal Broadcast (ACB) authenticated messaging service that we use over the P2P pub/sub dissemination channels

Decentralized Group Membership (DGA) protocol that establishes an eventually consistent membership set with causal ordering despite concurrent membership changes

Two-party Secure Messaging (2SM) end-to-end secure messaging protocol with PCS

Public Key Infrastructure (PKI) protocol for retrieving public key material and ephemeral pre-keys for group members

Decentralized Continuous Group Key Agreement (DCGKA) protocol for deriving keys for group members in response to messages received and membership change events, which keys are subsequently used for group message encryption.

Eventual consistency with causal delivery is a key building block for this protocol suite, as well as public key-addressed user and group identities.

Applied to the two-tier P2P setting, this protocol suite enables end-to-end secure communication channels directly between end-user devices, without proxies being able to decrypt application messages.

3.10 Edge networks

Nodes on LANs run the same set of protocols as the core network, but instead of using a peer sampling protocol for discovery that provides a partial view of the network, each node periodically announces its presence on the network by sending its public key to an IP multicast address reserved for this purpose. This allows nodes to construct a full view of the network by listening on this address for peer announcements. From this point on, the rest of the protocols are the same: the clustering protocol can use this full view of the local network to discover peers with overlapping subscriptions and join per-topic suboverlays.

References

[1] Alaggan, M., Gambs, S. and Kermarrec, A.-M. 2012. BLIP: Non-interactive differentially-private similarity computation on bloom filters. *Symposium on self-stabilizing systems* (2012), 202–216.

[2] Anceaume, E., Busnel, Y. and Sericola, B. 2013. Uniform node sampling service robust against collusions of malicious nodes. *2013 43rd annual IEEE/IFIP international conference on dependable systems and networks (DSN)* (2013), 1–12.

[3] Araujo, J.P. de 2019. *A communication-efficient causal broadcast publish/subscribe system*.

[4] Araujo, J.P. de, Arantes, L., Duarte Jr, E.P., Rodrigues, L.A. and Sens, P. 2019. VCube-PS: A causal broadcast topic-based publish/subscribe system. *Journal of Parallel and Distributed Computing*. 125, (2019), 18–30.

[5] Ariyattu, R. and Taïani, F. 2017. Filament: A cohort construction service for decentralized collaborative editing platforms. (2017).

[6] Beraldi, R., Quéma, V., Querzoni, L. and Tucci-Piergiovanni, S. 2007. TERA: Topic-based event routing for peer-to-peer architectures. (Jan. 2007), 2–13.

[7] Bortnikov, E., Gurevich, M., Keidar, I., Kliot, G. and Shraer, A. 2009. Brahms: Byzantine resilient random membership sampling. *Computer Networks*. 53, 13 (2009), 2340–2359.

[8] Chen, C. and Tock, Y. 2015. Design of routing protocols and overlay topologies for topic-based publish/subscribe on small-world networks. (Dec. 2015), 1–7.

[9] Donenfeld, J.A. 2017. WireGuard: Next generation kernel network tunnel. *NDSS* (2017).

[10] Eugster, P.T., Felber, P.A., Guerraoui, R. and Kermarrec, A.-M. 2003. The many faces of publish/subscribe. *ACM computing surveys (CSUR)*. 35, 2 (2003), 114–131.

[11] Frey, D., Guerraoui, R., Kermarrec, A.-M., Koldehofe, B., Mogensen, M., Monod, M. and Quéma, V. 2009. Heterogeneous gossip. *ACM/IFIP/USENIX international conference on distributed systems platforms and open distributed processing* (2009), 42–61.

[12] Frey, D., Jégou, A., Kermarrec, A.-M., Raynal, M. and Stainer, J. 2013. Trust-aware peer sampling: Performance and privacy tradeoffs. (2013).

[13] Guerraoui, R., Kuznetsov, P., Monti, M., Pavlovic, M., Seredinschi, D.-A. and Vonlanthen, Y. 2020. Scalable byzantine reliable broadcast (extended version). *arXiv preprint arXiv:1908.01738*. (2020).

[14] Jégou, A. 2014. *Harnessing the power of implicit and explicit social networks through decentralization*. Université Rennes 1.

[15] Jelasity, M., Voulgaris, S., Guerraoui, R., Kermarrec, A.-M. and Van Steen, M. 2007. Gossip-based peer sampling. *ACM Transactions on Computer Systems (TOCS)*. 25, 3 (2007), 8–es.

[16] Jesi, G.P., Montresor, A. and Steen, M. van 2010. Secure peer sampling. *Computer Networks*. 54, 12 (2010), 2086–2098.

[17] Martori, J. and Urso, P. 2016. *Reliable causal delivery with probabilistic design*. Technical Report #RR-8985. INRIA Nancy.

[18] Matos, M., Nunes, A., Oliveira, R. and Pereira, J. 2010. StAN: Exploiting shared interests without disclosing them in gossip-based publish/subscribe. *IPTPS* (2010), 9.

[19] Matos, M., Schiavoni, V., Riviere, E., Felber, P. and Oliveira, R. 2014. LayStream: Composing standard gossip protocols for live video streaming. *14-th IEEE international conference on peer-to-peer computing* (2014), 1–10.

[20] Nagy, M., Cristofaro, E. de, Dmitrienko, A., Asokan, N. and Sadeghi, A.-R. 2013. Do i know you? – efficient and privacy-preserving common friend-finder protocols and applications. Cryptology ePrint Archive, Report 2013/620.

[21] Patel, J., Rivière, É., Gupta, I. and Kermarrec, A.-M. 2009. Rappel: Exploiting interest and network locality to improve fairness in publish-subscribe systems. *Computer Networks*. 53, (Aug. 2009), 2304–2320. DOI:https://doi.org/10.1016/j.comnet.2009.03.018.

[22] Perrin, T. 2018. The noise protocol framework. (2018).

[23] Prakash, R., Raynal, M. and Singhal, M. 1997. An adaptive causal ordering algorithm suited to mobile computing environments. *Journal of Parallel and Distributed Computing*. 41, 2 (1997), 190–204.

[24] Savolainen, P., Juslenius, S., Andrews, E., Pokrovskii, M., Tarkoma, S. and Pihkala, H. 2020. The streamr network: Performance and scalability. (2020).

[25] Setty, V., Steen, M. van, Vitenberg, R. and Voulgaris, S. 2012. PolderCast: Fast, robust, and scalable architecture for P2P topic-based pub/sub. (Dec. 2012), 271–291.

[26] Weidner, M., Kleppmann, M., Hugenroth, D. and Beresford, A.R. 2020. Key agreement for decentralized secure group messaging with strong security guarantees. Cryptology ePrint Archive, Report 2020/1281.