Decentralized Hashtag Search and Subscription for Federated Social Networks

@schmittlauch@toot.matereal.eu

22.12.2018

Motivation

With the final standardization of the ActivityPub[1] protocol by the W3C, federated social networks seem to have gained traction again with several new social servers implementing it[2].

But even these current implementations still suffer from a limitation all social networks based on push-federation still have: They cannot provide a consistent network-wide view on all public posts including a certain hashtag. Such a search currently only returns posts from the instance's[3] local database which have been delivered to the instance anyways due to other subscriptions. This limits the user experience compared to centralized mainstream social networks like Twitter, Tumblr or Facebook.

For the Diaspora* network there exists an implementation of a centralized federation relay, where federation servers send all their public posts to and can register to receive all messages containing a certain hashtag. But this centralized approach is against the spirit of federation and decentralization as it forms a single point of failure and potential bottleneck.

Planned Work

The push-federation principle of current federated social networks works on the basis that user identifiers always include their home instance. Thus the server responsible for managing subscription requests and delivering new posts to all subscribers is always known. But there is no such place for all messages containing a certain hashtag, as these can originate from any instance. In federated social networks it is even not necessary for all instances to know each other.

I plan to provide such responsible subscription points as an additional mechanism for ActivityPub-compatible federation servers:
All participating instances form an overlay network using a Distributed Hash Table (DHT) as an addressing and routing mechanism. Responsibility for a certain hashtag is then distributed over the DHT's key space to the closest n nodes. This group of nodes is then the responsible point for managing subscriptions. Participating nodes send all messages containing a certain hashtag to these responsible nodes, which then forward the posts to all subscribing instances. The DHT is only used for delegating and discovering responsibility for hashtags, while the post delivery happens directly routed between instances and subscription nodes using normal ActivityPub push mechanisms.

Additionally to this subscription and relay mechanism I want to provide a method for instances to query posts under a hashtags even in retrospective and without subscribing. The reasoning behind this is that users should be able to first get an impression of a hashtag's post history before deciding to subscribe to it, and also be able to get posts published before their instance subscribed to that hashtag.
For this, the nodes responsible for a hashtag also need to store the history of posts containing their hashtag and provide all posts from a requested time frame to a requesting instance.

Although a network of server instances will not have a high fluctuation of nodes, measures need to be taken to provide some redundancy of responsible nodes per hashtag. There also needs to be a mechanism for transfering the stored message history to

[1] http://activitypub.rocks

[2] Mastodon, Pleroma, PixelFed are examples for general social networks, with several special-interest software like FunkWhale adopting the protocol as well

[3] a federation server domain

new nodes.

One possible problem that might arise is balancing the load of "hot-spot" nodes responsible for popular hashtags. Real-world data has to be analyzed to estimate whether load balancing and storage distribution mechanisms are necessary even within a hashtag.

The system performance shall at least be able to handle the throughput present in existing centralised networks.
For Twitter this means dealing with a global message rate of averagely 1620 posts per second, with peaks going up to 143,199 posts per second[4]. But only an eventually-consistent view is needed, posts are allowed to need up to a few minutes to propagate through the network.
These performance requirements are going to be evaluated at least by calculations or even by simulations of such a system. It needs to be evaluated whether relaying only references to a post creates a sufficient performance, or whether relaying the posts themselves is needed.

Security Objectives

As this approach shall only be used to federate public posts, confidentiality is not a big issue. It is worth to be evaluated though how good deletion/ recall of posts can work in such a network.

Posts are signed like in current ActivityPub implementations for providing a way to verify their integrity. But the more important aspect of integrity is that an attacker must not be able to deliberately gain the responsibility for a certain hashtag and then silently drop messages. The integrity of the message history itself needs to be ensured.
At a further look, depending on the stored content a way to transparently exclude messages from a node's store due to legal reasons might be necessary.

[4] https://blog.twitter.com/official/en_us/a/2011/numbers.html