Mikael Blomstrand (mbloms@kth.se), Filip Johansson (fij@kth.se)

Project Plan IK 2218 - The rumor protocol

The rumor protocol is a decentralized peer-to-peer gossip protocol that spreads rumors from peer-to-peer in the network. The idea is that it should be minimal and general purpose, rather than addressing a specific use.

Functionality

The Rumor Protocol should have the following functionality. Peers should periodically send IP addresses and data to peers in the network.

Peer Exchange

A peer in the network should be able to send IP addresses from its IP list to any other peer in their list.

Send Data

A peer should be able to send data to any other peer in its list of peers. A hash of the data is included to ensure data integrity.

Request Data

A peer should be able to request data from any other peer in its list of peers.

Data

Data will be stored in a hash table with (key, value) where key is a hash of the data and value is the data.

Technology

- The protocol will be based on UDP.
- All communication should be simplex.

Possible extensions

- Group peer exchange, send data and request data together in one datagram.
- Onion routing.
- Signatures instead of hashes.

Possible usage

- Share git objects
- Twitter like social network (GUI)

Requirements and Testing

Requirement 1:

When introduced to a new peer (by receiving an IP-address from a known peer, or by being contacted by an unknown peer) a client should be able to store this information and use it in succeeding communication.

Requirement 2:

When receiving data, a client should be able to store this data so that it can be retrieved later.

Requirement 3:

When receiving a request, a client should be able to send the data back if it has it.

Concrete test scenario 1:

In a network with hosts A,B and C. A has the IP's of B and C. A sends some data to B. A sends C's IP to B. After a delay or continuously A requests the data from C.

B should be able to store the data and A's IP. When receiving C's IP B should be able to use this information to send the data to C. C in turn should be able to store the data and B's IP. C should be able to send the data to A when receiving a request for it.

Concrete test scenario 2:

In one network with 10 hosts arbitrarily connected into a single component. This could be achieved by manual configuration or some implementation feature, but a star network should be avoided. Hosts should start with at most two peers each. One host sends some data eg. tweet-like text message. The data should start spreading to all nodes in the network, and peers should start exchanging peers. Eventually the data should have spread to all nodes.

In order to evaluate performance and behaviour (such as overhead and redundancy) all clients should log all traffic with time stamps.

Interesting parameters to evaluate:

Time until each peer receives the data. How many times the data was sent/received in total before spreading to all nodes. How IP's (peer relations) propagate in the network. How long time until the network resembles a full graph.