Ensichat Implementation Concept

Felix Ableitner

March 27, 2016

1 Topology

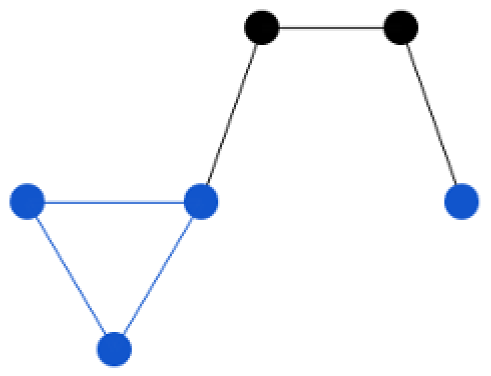

Figure 1: A possible network topology. Black nodes are servers, blue nodes are smartphones. Blue edges are Bluetooth connections, black edges internet connections. Note that connections can be opened or closed at any time.

The network consists of two different node types, smartphones and servers. Both use the same protocol, and run the exact same code. The protocol does not differentiate between the different node types in any way.

All nodes can be connected over the internet, if a connection is available. Additionally, smartphones will connect to each other via Bluetooth, if they are in range.

Servers are not required for the protocol to work. They are only used as an additional transport mechanism, to help if nodes are too far away from each other for a Bluetooth connection.

Every node has a message buffer, containing messages that have yet to be delivered. The buffer has a size limit, where sensible defaults might be 250 MB for phones and 5 GB for servers. Messages are sorted by time received. If the buffer is full, the oldest message will be deleted. Messages that were sent by the local node are only deleted from the buffer after the message is confirmed delivered.

2 Message sent from Local Node

To ensure fast and certain delivery, we send multiple copies (forwarding tokens) of each message. To simplify the initial implementation, we set a hardcoded value of 3 forwarding tokens. This assumes a total number of around 100 nodes. It will have to be changed if the network size changes significantly.

At a later point, we can implement formulas so that each node can calculate the number of copies itself, as described in **(author?)** [3].

3 Spray Phase

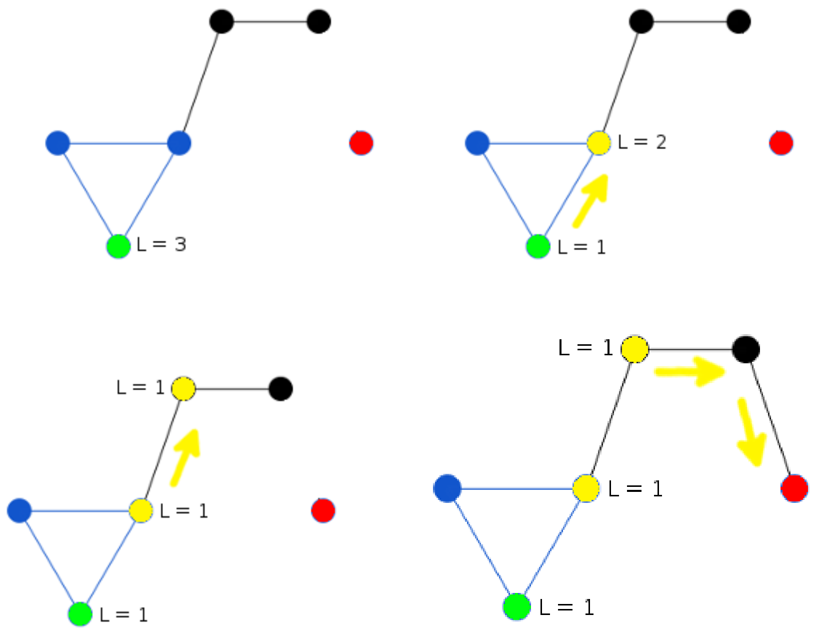

Figure 2: Green sends a message to red. Yellow are the relays.

Relays are chosen among the direct neighbors of a node. We pick relays using a utility function, as described in **(author?)** [2]. As our utility function, we use the overall time we were directly connected to each node in the network. This means, whenever we connect to a node n, we start a counter t, and stop

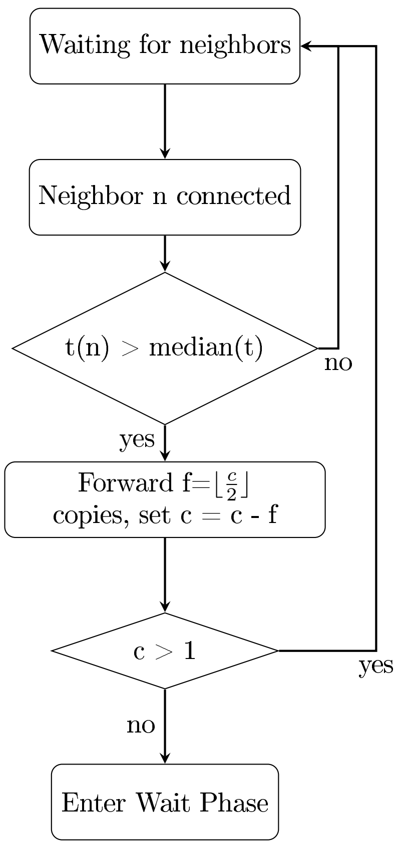

Figure 3: Flowchart of a node in the spray phase for a given message.

the counter when we disconnect. The counter increments every second, and is continued from the last value when we encounter the same node again.

The intuition behind this is that nodes with longer connection time are better connected in general. This is especially true for internet servers, which are connected to many nodes, and should be picked as relays first. At the same time, we don't have to rely on data that nodes report themselves, keeping the network trustless. Our utility function is then $t(n) > median(t)$. If a neighbor fulfills this formula, we choose it as a relay.

If a node is chosen as a relay, we forward $\lfloor \frac{c}{2} \rfloor$ tokens to it (c is the number of tokens we currently have), and keep the rest of the tokens. When a relay only has one token left, it enters the "Wait" phase.

10^1s	~10s
10^2s	~2m
10^3s	~17m
10^4s	~3h
10^5s	~28h
10^6s	~12d

Figure 4: Approximate values for w.

4 Wait Phase

In the wait phase, relays use AODVv2 **(author?)** [1] to try and find a route to the message destination. With the default settings, AODVv2 will only retry a failed route discovery for approx. 10 seconds, and discard the message after this time. To ensure a high delivery ratio, we need to increase this duration. At the same time, we have to limit the retry interval so we don't overload the network. We define a function that calculates the number of seconds w to wait before the nth retry, measured in seconds since the (n-1)th retry: $w = 10^{min(n,6)}$. This formula is chosen somewhat arbitrarily, and a different formula might prove more efficient in real-world use cases.

If the buffer contains multiple messages for the same destination, only the waiting time for the newest message will be considered for route discovery. If a route is found, all messages for this destination can be delivered at once. This reduces resource usage in case a lot of messages for the same destination are relayed by the same node.

In addition to the active route discovery, a node will also deliver the message if it passively receives a route to the destination via AODVv2, or if it connects directly to the destination node.

Each relay tries to deliver its message copy independently. This means that a destination will likely receive multiple copies of a single message. In this case, any additional copies will be ignored by the destination node.

5 Reliability

Due to the number of independent relays, it is very likely that every message will be delivered soon. However, delivery can fail if none of the relays can reach the destination. This can happen for any of the following reasons:

- Relays are taken offline for any reason
- The relay buffer overflows, so the message is deleted
- The destination node does not connect to any relay's mesh network

However, even if all relays fail, the message can still be delivered by the original sender. As discussed earlier, the sender always keeps his own messages in the buffer, until they are confirmed delivered.

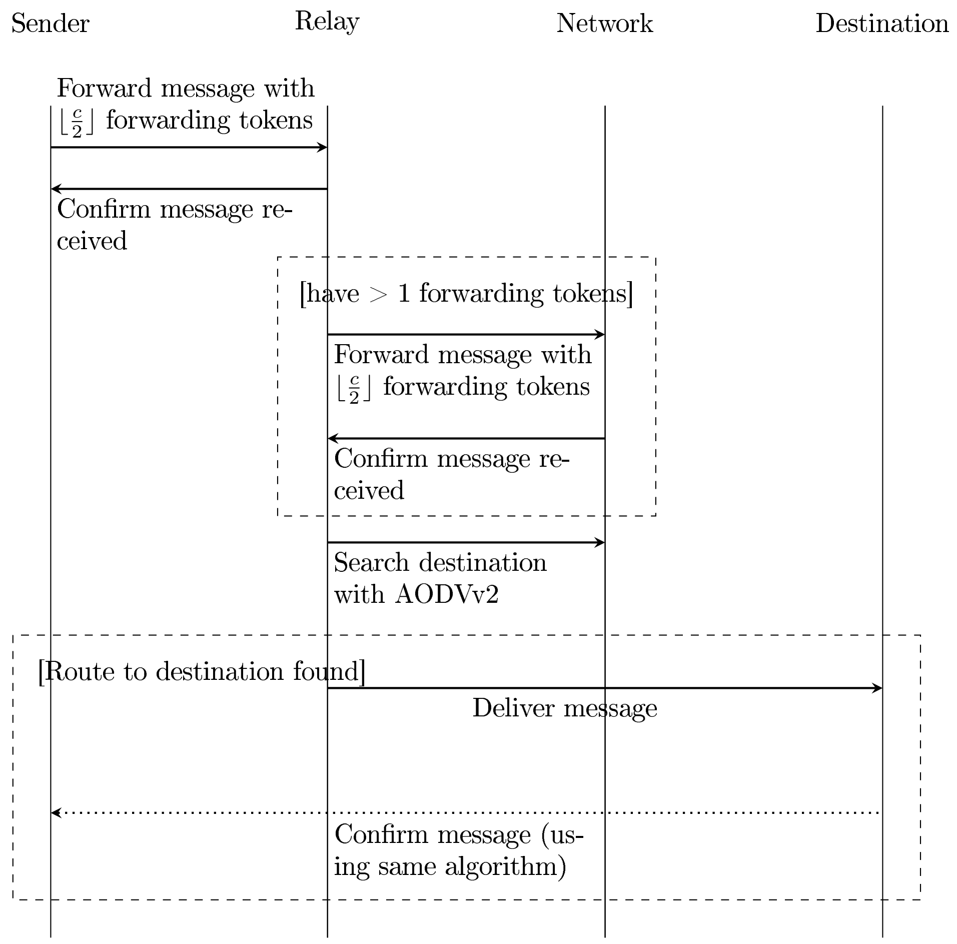
Sender
Relay
Network
Destination
Forward message with $\lfloor \frac{c}{2} \rfloor$ forwarding tokens
Confirm message received
[have > 1 forwarding tokens]
Forward message with $\lfloor \frac{c}{2} \rfloor$ forwarding tokens
Confirm message received
Search destination with AODVv2
[Route to destination found]
Deliver message
Confirm message (using same algorithm)

Figure 5: Message transmission between devices. Note that the sender performs the same discovery actions as the relay. Additional relays are not considered here.

Once the message arrives at the destination node, it will send a confirmation message back to the original sender (but not to the relays). The same routing algorithm is used to send the confirmation message.

6 Security Analysis

Like in any decentralized network, a sybil attack is always possible. That means, an attacker can launch many nodes, and ensure that a node is only connected to malicious nodes, which don't forward any messages (making this a denial of service attack).

As another attack, a malicious actor can try to flood the buffers of nodes, so that legitimate transactions are deleted. As a countermeasure, potential relays should make sure that the number of forwarding tokens cannot be set arbitrarily high for a single message.

But even then, an attacker can just send many messages, possibly from many different nodes. To defend against this, a proof of work algorithm may be used, or micropayments from the sender to the relays.

However, a detailed threat analysis or defensive measures are not part of this thesis.

References

[1] Mobile Ad hoc Networks Working Group. Ad hoc on-demand distance vector version 2 (aodvv2) routing, 2016.

[2] Thrasyvoulos Spyropoulos, Thierry Turletti, and Katia Obraczka. Routing in delay-tolerant networks comprising heterogeneous node populations. *Mobile Computing, IEEE Transactions on*, 8(8):1132–1147, 2009.

[3] Spyropoulos Thrasyvoulos, Konstantinos Psounis, and Cauligi S Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking. ACM*, 2005.