API Sketch

Freitag, 11. November 2016 18:02

Nodes : N_i $i := 1, 2, \dots, n$

Messages : m_i

Last Contact Table :

N_1	18:07
N_2	18:00
⋮	⋮
N_n	13:33

Each node has a LC-table with the others nodes names and a timestamp stored.
The timestamp represents the time when the node was last present in the network. That means that the timestamp is updated each time you hear from another node or you get an LC-table from another node.

ACK-Table :

I: What happens with two messages at a time since the table can only display 0 or 1??

Receiver

Sender	N_1	N_2	...	N_n
N_1			...	
N_2			...	
⋮			⋱	
N_n			...	

The ACK-Table describes which Receiver nodes got a message from a particular Sender node. The green parts are 0 for "no message received" or 1 for "message received". Important to mention is, since the API is based on a decentralized system, that the table only shows the view seen by the owner of the table at each time.

Message :

origin node which sent the msg

Last Contact Table
ACK - Table
Content

broadcast_message(origin, msg) : 1. call send_message for each reachable node except of those which got the message already according to the ACK-table.
Handle limitations of WiFi Direct on # connections

send_message(origin, msg, receiver) : 1. Connect with node over Wi-Fi Direct
2. Send message sent on the left
3. Wait for acknowledgement → timeout → repeat step 2
Handle loss of connection at any time
4. Update own ACK-table and change entry (origin, receiver) to a one.

node which gets the msg

ACK :

Last Contact Table
ACK-Table

receive_message : 1. Send acknowledgement
2. Update ACK-table from message with own (change each entry which is in at least one of the tables 1 to a 1)
3. Update LC-table from message with own (take the earlier timestamp for each node)
4. If all nodes got the message ⇒ do nothing
else ⇒ broadcast_message(...)

II: Better way?
If nothing changes both nodes have to check both tables and possibly update them.

III: Or what?

table_exchange : 1. ~~~
2. ~~~
3. ~~~
4. ~~~