

Fig. 4. Reputation checking during the publication process

B. Other Services

When a peer is connected to the network, services (publication of contents, search, data download...) are achieved by sending requests to the other peers. In Kademlia [13] this is done in two phases. Firstly, *Kademlia_REQ* are sent to find nodes which are potentially able to deliver the service (according to their place in the DHT). This phase is general and only concerns the iteration mechanism used to find several peers in a part of the P2P network. In the second step, when the nodes are found, a specific request is sent to ask for a particular service. The reputation checking must be done before the specific request for three reasons. Firstly, a peer can be a bridge and search contacts for other uncontrolled peers. Secondly, the real services are provided by the specific requests. Controlling the reputation at this point allows to revoke independently the different services. Finally, checking the reputation for *Kademlia_REQ* would increase the overhead for no advantage. The figure 4 presents the running of a publication request and includes the reputation checking. The other requests (search, data download) follow the same scheme.

However, inserting the revocation mechanism into the search function is not relevant for several reasons. Firstly, it is not a service through which a peer can damage the network. Then, searching is useless when the other services are inactive behind. Finally, it would also introduce overhead to the network and unnecessary delay for all users.

C. Implementation

We have implemented the revocation mechanism in KAD. To do that, we have introduced different modifications in the KAD client:

- our modified client can manage a new kind of information called "Account";
- the associated requests search/store Account were written, which partially behave like existing requests on keywords, files or notes;
- new functions were added in the UDPListener, where all network's requests are processed. In fact, these functions do the service-oriented revocation, searching for the reputation and checking it.

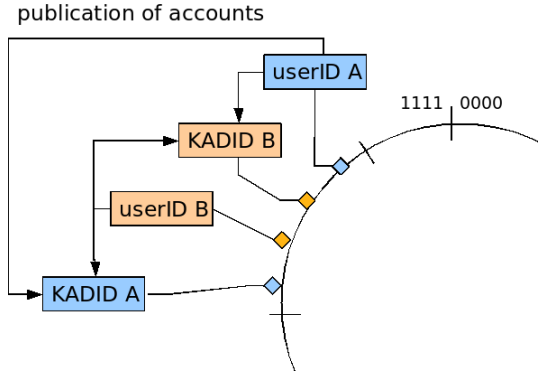


Fig. 5. Publication of accounts between modified KAD clients inside a tolerance zone

- KadID and userID allocation were cheated to control the place of the peers in the DHT.

We have tested that the reputation storage and retrieval and the revocation of services work fine on the modified peers. However, as our implementation defines new data types and messages to manage the accounts, we can just test our revocation mechanism on a few peers. Presently, as we test the mechanism with few resources, we have defined the KadID and UserID of modified peers to place them in the same tolerance zone. In this way, we are sure that the accounts can be stored (figure 5) when using the KAD publication mechanism. That is enough to verify the mechanism, but without the real number of replicated peers, performance measures would be wrong. That is why we are going to scale up our testbed on EmanicsLab to measure performances and compare the results with the evaluation presented in section V.

V. ANALYSIS AND DISCUSSIONS

A. Performances Evaluation

The thesis [2] has led a performance evaluation of the KAD network which allows us to discuss some a priori performance results. The average delay needed to store information in the network is about 200 seconds. This time is needed to find ten peers (for the replication) with a KadID close to the hash of the information to store. This delay will occur the first time that a peer connects itself to the network and periodically later to maintain the account in the network despite the churn. The delay to retrieve the information fluctuates and depends on the replication. The more replication there is, the more robust the stored information is and the quicker it is retrieved. The information is retrieved linearly, so the more a peer waits, the more results are returned. A 100 seconds delay seems to be sufficient to retrieve enough information to guess the real reputation of a peer. This delay could seem huge because it appears prior to each service of the network, except the search as explained. But in fact, 100 seconds to bootstrap are not penalising, the publication process is entirely transparent for the user and 100 seconds preceding a download