DRSTL weekly progress Report I

Hebi Li

11/25/2013

Table of Content

ChronoSync, a recent published method solving this proble

Traditional Methods

DRSTL design

Comparation and merits

ChronoSync Review

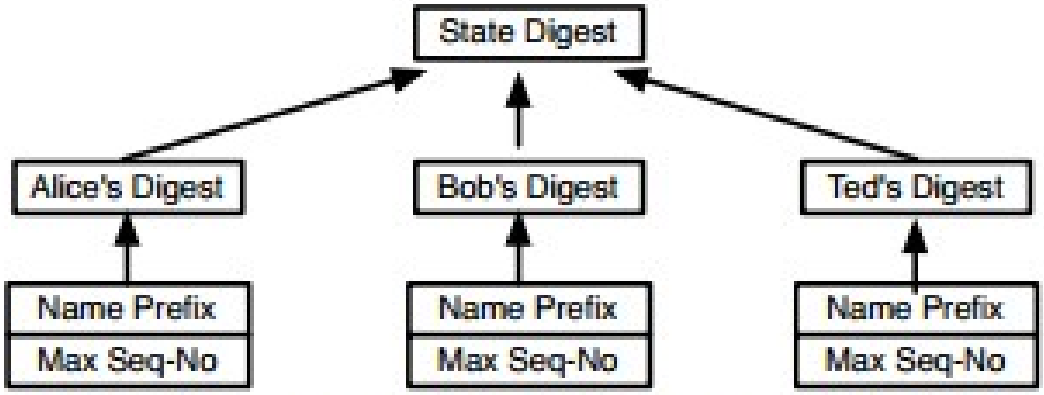
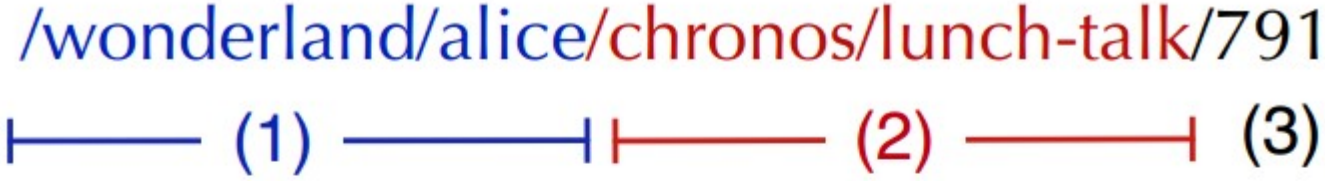


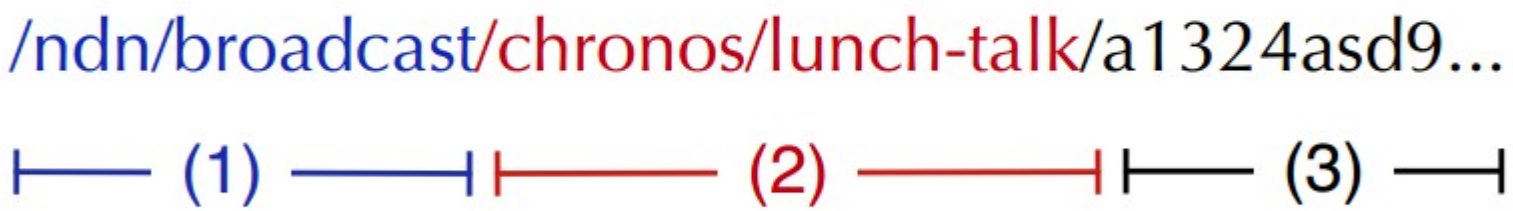
Fig. 4: An example of digest tree used in ChronoChat

State Digest	Changes
0000...	Null
9w35...	[Alice's prefix, 1]
...	...
23ab...	[Bob's prefix, 31], [Alice's prefix, 19]
05t1...	[Bob's prefix, 32]

TABLE I: An example of digest log



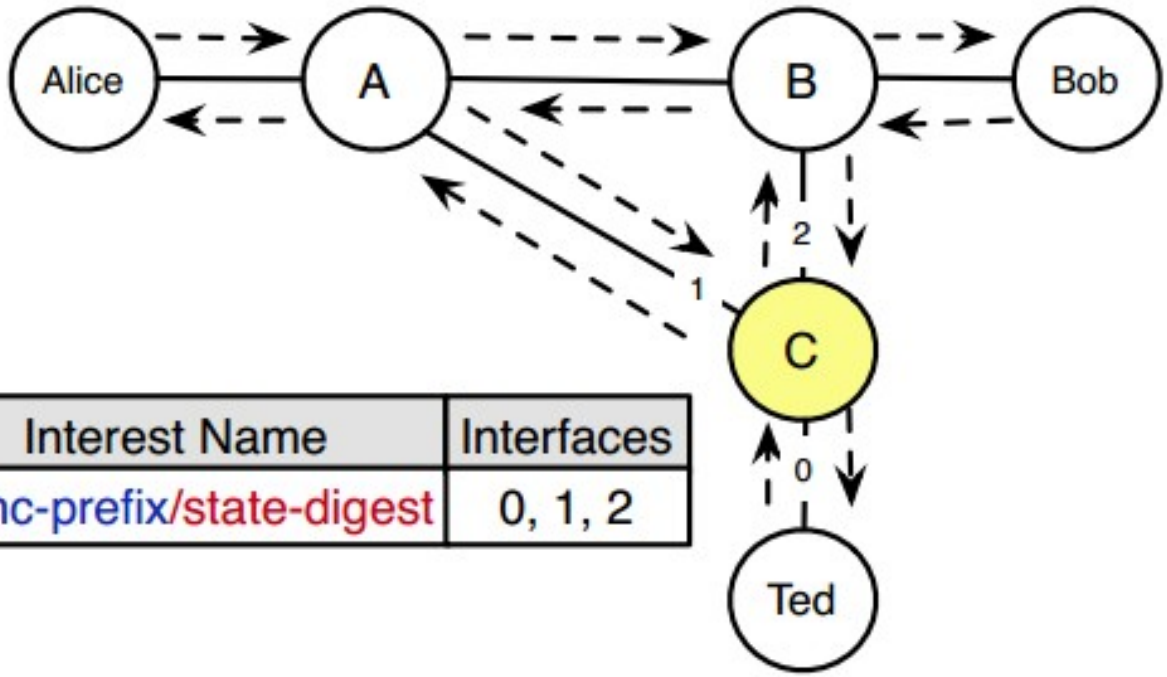
(a) An example of chat data name



(b) An example of sync data name

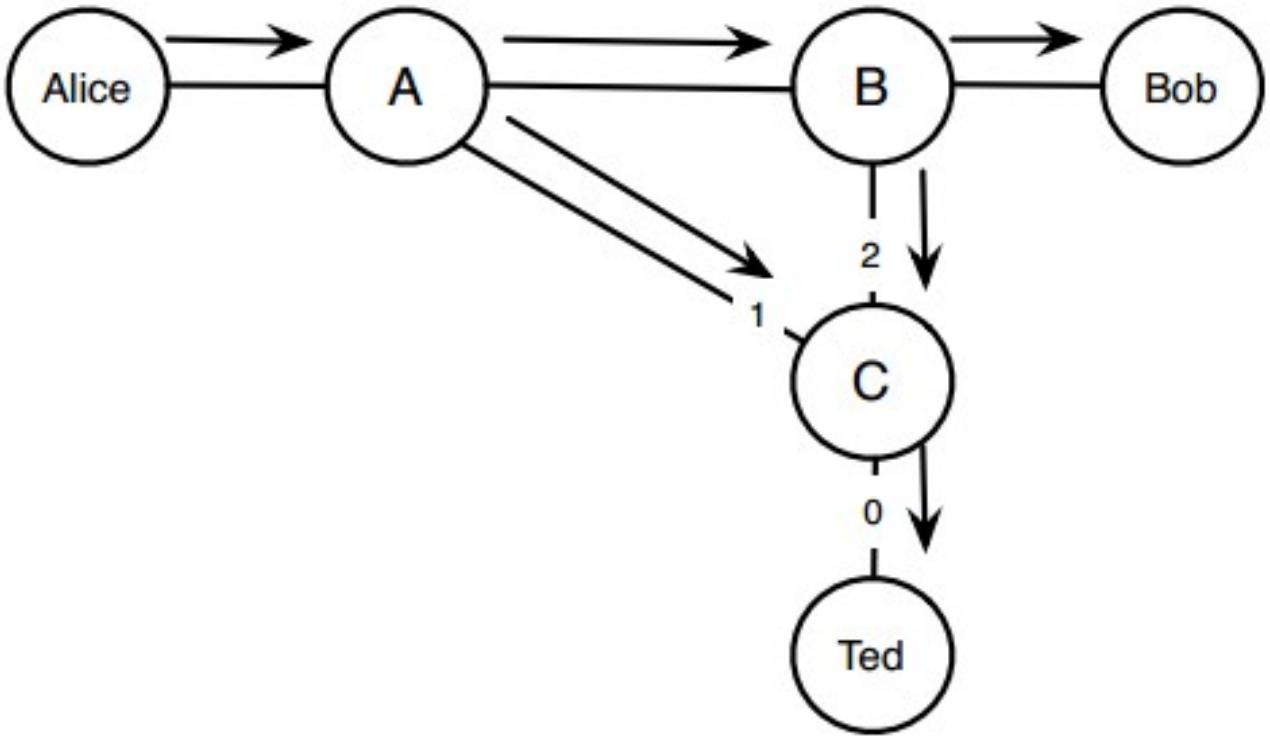
Fig. 3: Naming rules of ChronoSync

- routable name prefixes
- identifies the process that is responsible for handling such interests
- last part carries the latest state digest of the interest sender



Interest Name	Interfaces
/sync-prefix/state-digest	0, 1, 2

(a) For each chatroom, at most one sync interest is transmitted over a link in one direction. There is one pending sync interest at each party in the stable state. Due to space limit, only the PIT of router C is depicted.



(b) The state change caused by *Alice*'s new message is multicasted to other two users following the PIT entries set up in routers by sync interests

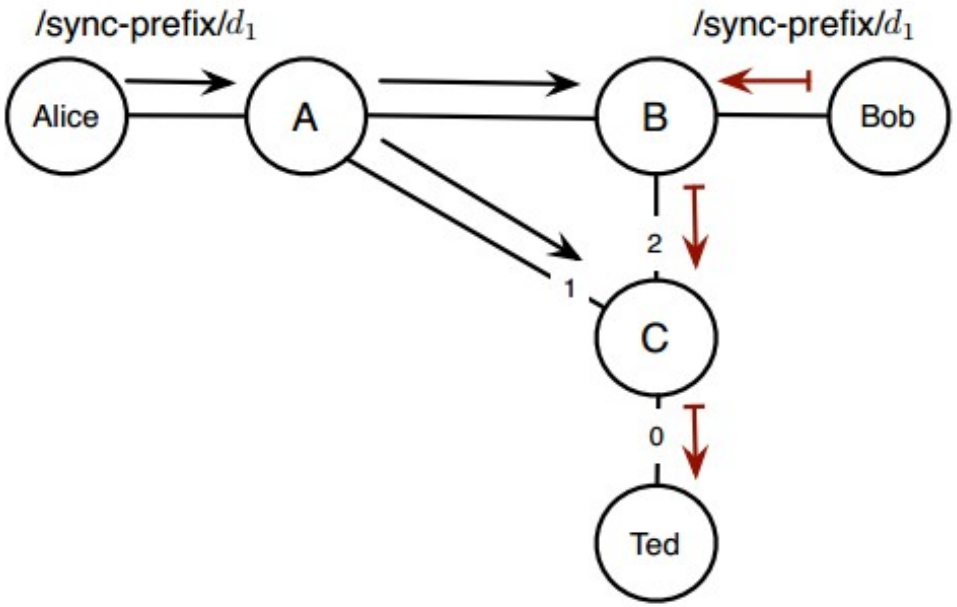


Fig. 6: An example of simultaneous data generation

send a sync interest with the previous state digest again, but this time with an exclude filter that contains the hash of Bob's sync data

If there are more producers involved in a simultaneous data generation event, multiple rounds of sync interests with exclude filter have to be sent. Each of such interest has to exclude all the sync data of a particular state digest known to the requester so far.

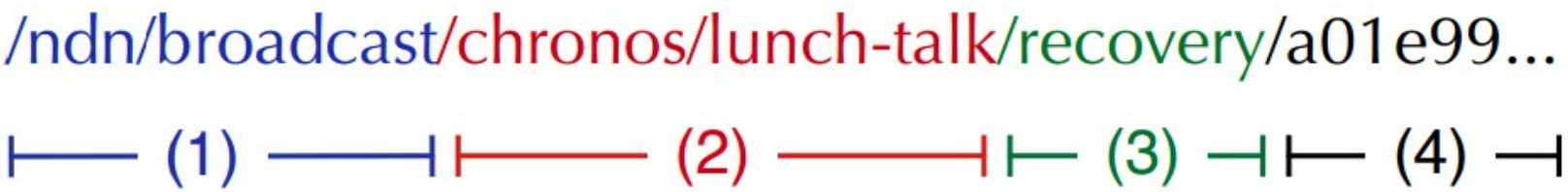


Fig. 7: An example of recovery interest

The recipient of the unknown digest sends out a recovery interest

Those who recognize the digest (e.g., having it in their digest log) reply the recovery interest with the most recent producer status of all users

- ChronoSync uses the broadcast sync interests to exchange state digests. However, broadcasting sync interests to the parties scattered in large networks, such as the Internet, could be costly.

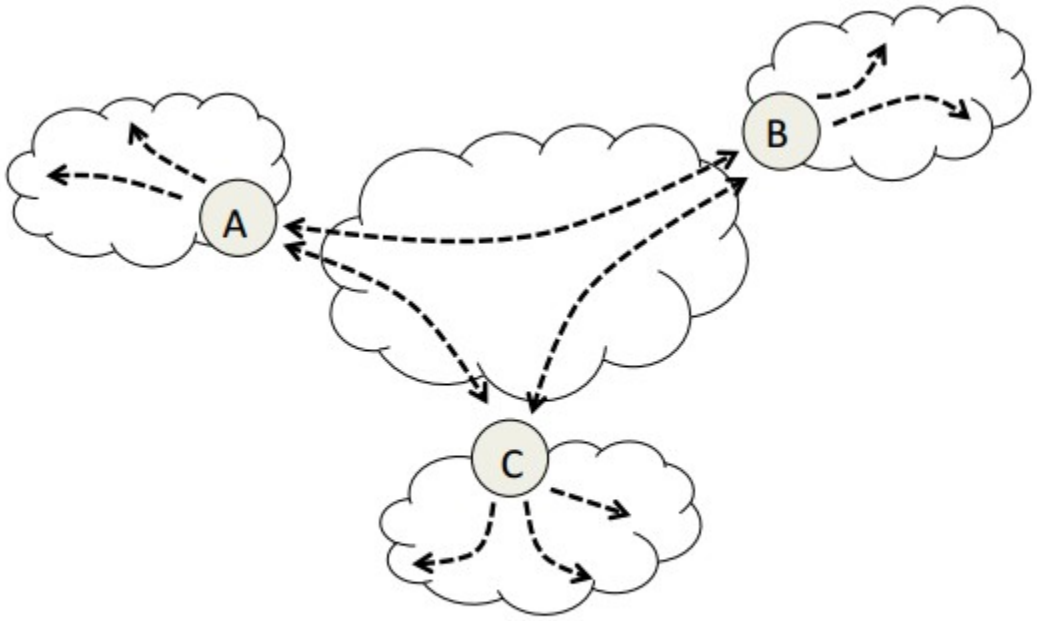


Fig. 17: Overlay broadcast network for ChronoSync

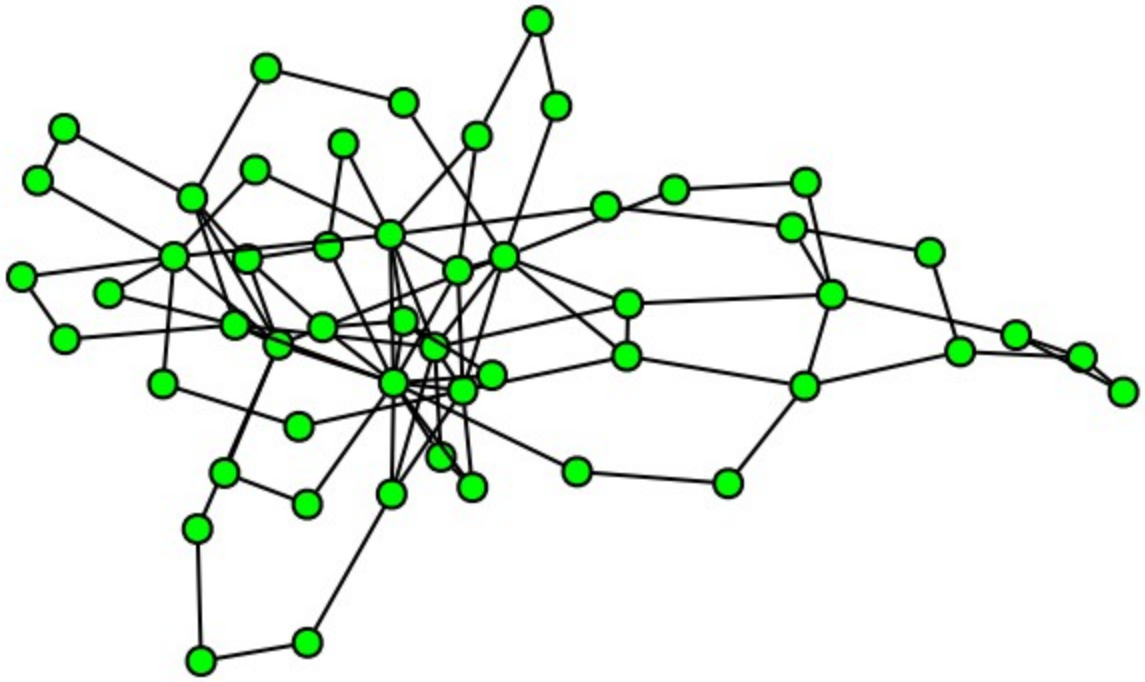
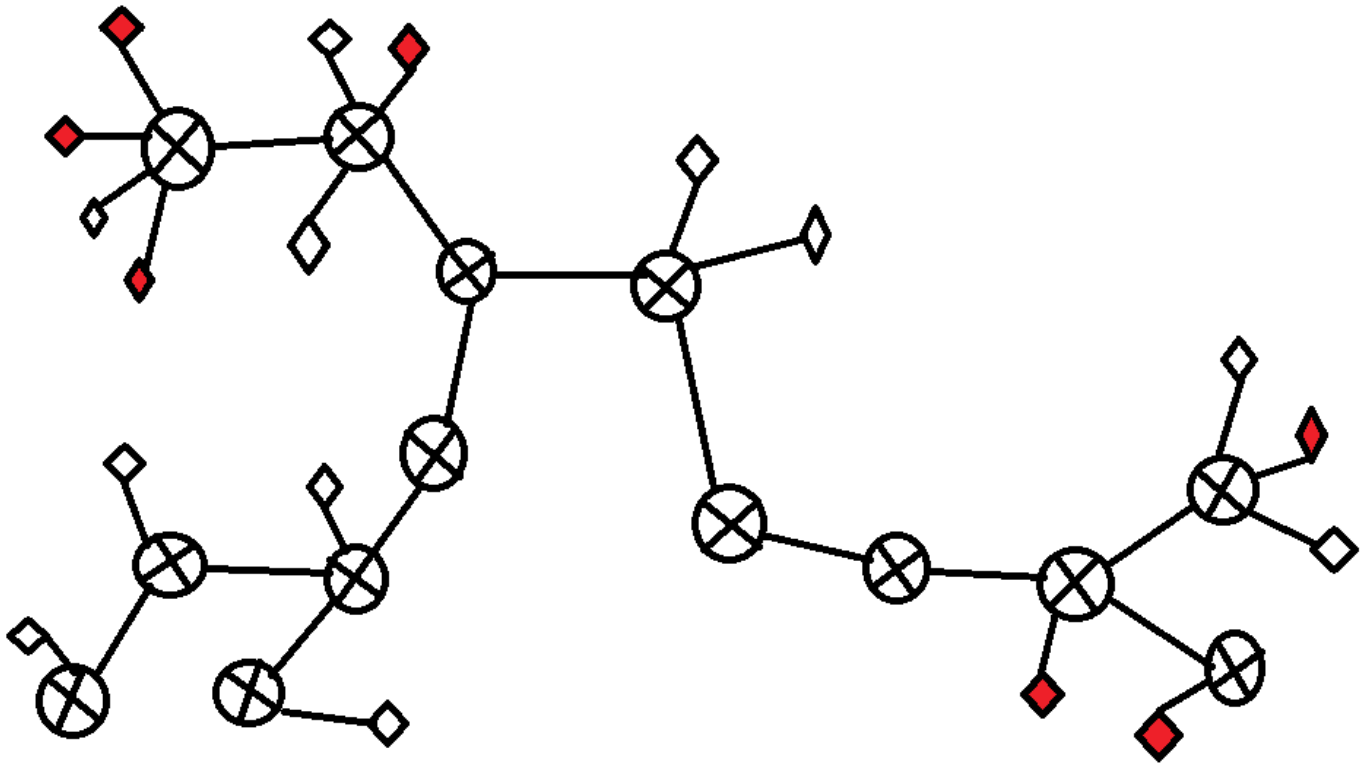


Fig. 9: Sprint point-of-presence topology



Has Problem with Broadcast Sync Interest.

DRS: Distributed Random Server for NDN Synchronization

- Distributed Random Server is a spontaneous server within a range of local area to handle synchronization jobs in its whelm.
- All the autonomous groups form a group layer.
- Beyond this layer, it follows the similar steps to build the higher level groups with a higher level server selected randomly as well.
- Thus the synchronization work could be divided into small pieces on multi levels, an efficient way with scalability in large networks.

Traditional Server

- there should be large latency even if two nodes are near each other
- the usage of links is un balanced.
- Not robust

ChronoSync

- the outstanding sync interest should be broadcast and every links in this system should get one interest copy, so as the data
- Scalability: gateway
  - damage in its distributed design
  - damage the robustness: Although participants could communicate within the region without any affect, but that's it, they still can not talk to the guy outside
  - will there be such gateway?
- Complicate and inefficient to handle trouble:
  - simultaneous data generation exclude filter
  - Physical divide recovery interest

Naming rules

I	/ndn/broadcast/chatapp/anyserver/seq
~~D~~	~~/ndn/data/chatapp/~~iamserver/seq/alice
I	/alice/chatapp/iamclient/seq/bob
~~D~~	~~/ndn/data/chatapp/~~registered/seq/bob

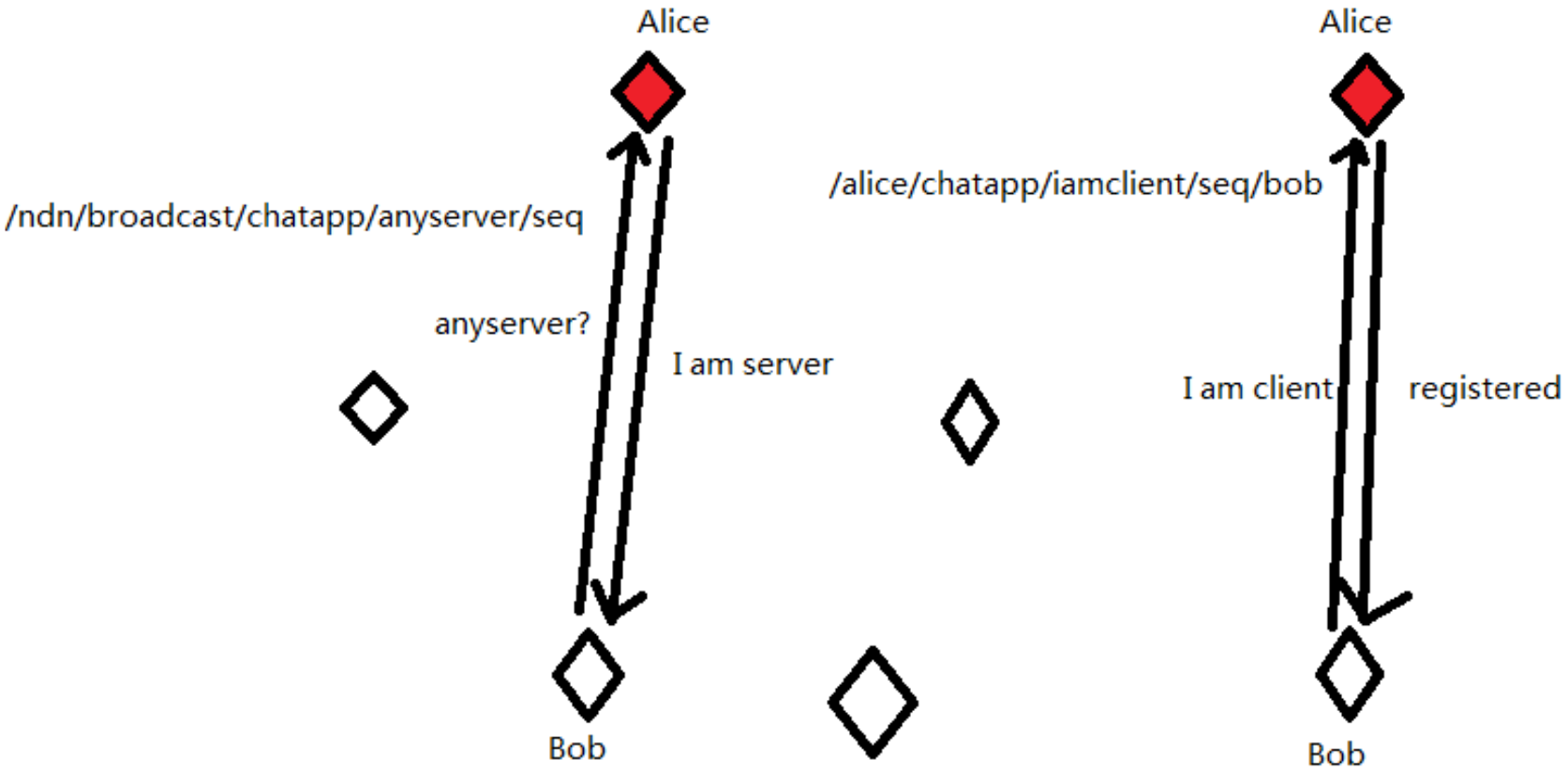
Table 1: server generation naming

I	/alice/chatapp/anythingnew/seq/digest
~~D~~	~~/ndn/data/chatapp/~~newthing/seq/digest/msg
I	/alice/chatapp/somethingnew/seq/digest/msg
~~D~~	~~/ndn/data/chatapp/~~acknowledged/seq/bob

Table 2: sync naming

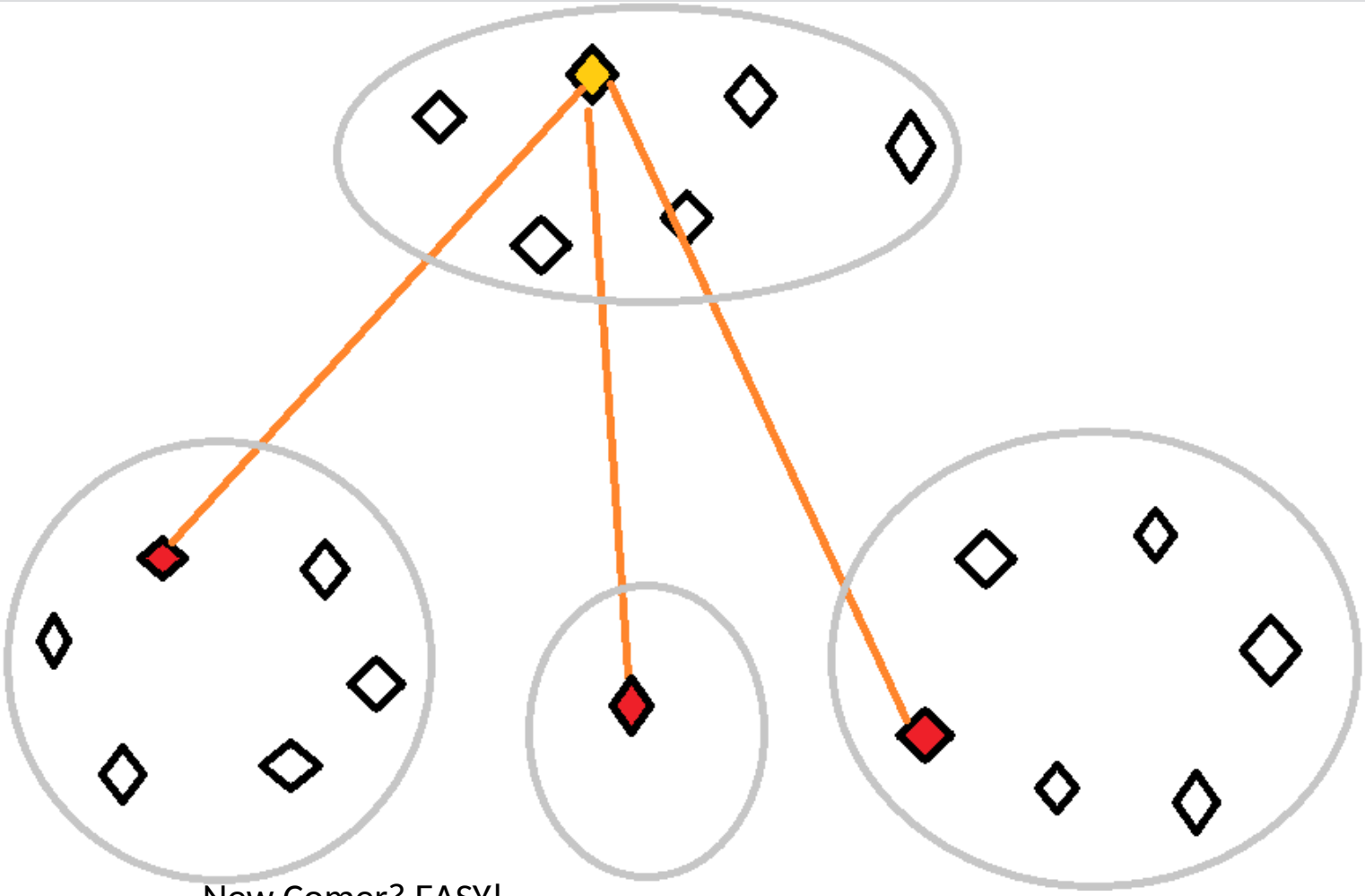
Server generation

- Every participant who does not have a server should wait for a random time before he broadcasts out the interest requiring is there are any servers nearby
- After the interest timeouts, the participant is ready to send ’iamserver’ data packet back to the same interest from others


Alice
/ndn/broadcast/chatapp/anyserver/seq
anyserver?
I am server
Bob
Alice
/alice/chatapp/iamclient/seq/bob
I am client
registered
Bob

Multi-level architecture

- Every interest and data related should have a seq indicating it is on which level
- If the one participant don’t receive a client’s confirm, he should assume he’s the topest level server.

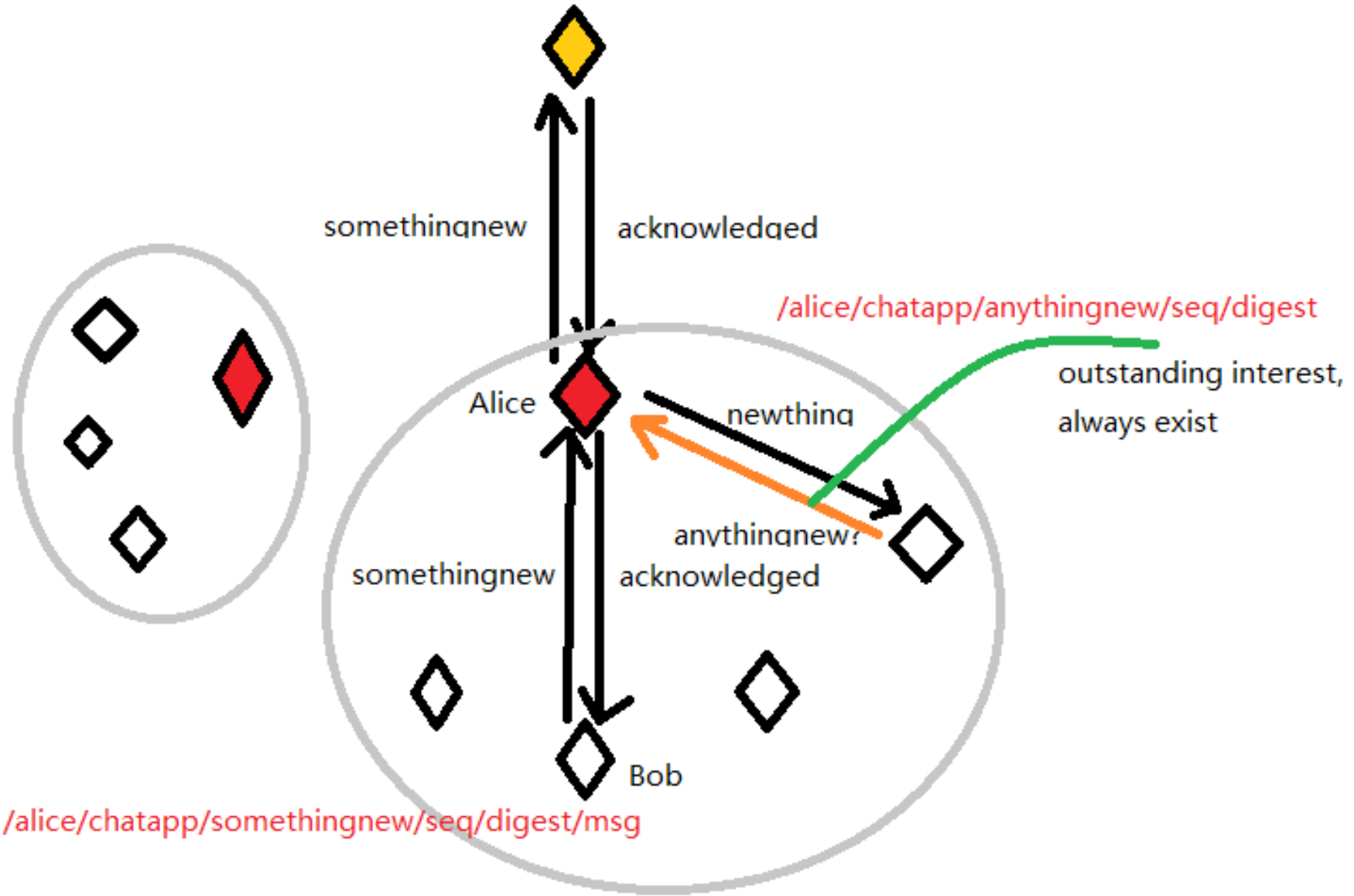


New Comer? EASY!

Server fail? Re-select!

Synchronization

- When some one in a group generate a new message, he will first update his own state digest. After that, he will send the new digest, the old digest, as well as chat message itself by a *somethingnew* interest.
- Every members in a group will continuously send *anythingnew* interest to the server


somethingnew
acknowledged
/alice/chatapp/anythingnew/seq/digest
outstanding interest,
always exist
Alice
newthing
anythingnew?
somethingnew
acknowledged
Bob
/alice/chatapp/somethingnew/seq/digest/msg

Merits

- Robust
- Distributed
- Scalability
- Direct and Neat

Any Suggestion?

- NDN broadcast
- Must implement and evaluate?