The Security and Dependability of Distributed Systems- Spring 2017

Final Course Project-

BeerClan : A Decentralized Online Social Community

Under supervision of: Professor. Naftaly H. Minsky

Saurabh Singh
NetID- ss2716
Email: s.singh@rutgers.edu

Alok Singh
NetID- as2509
Email: alok.s@rutgers.edu

Date: May 4, 2017

BeerClan : A Decentralized Online Social Community

Overview: BeerClan is an online social community for beer and wine enthusiasts to connect to each other and discuss their interests. In more formal terms, it is a Decentralized Online Community that attempts to mitigate the risks and privacy concerns of a conventional or centralized Online Social Community. BeerClan uses Law Governed Interaction (LGI) as middleware for access control. The model successfully implements complex social interactions and relationships like following, friends, controlled personal messaging, posts, invitation, approval, removal, interests & narrow-casting. What follows in this report is a detailed description of each of these interactions/relationships and the strategy we use to implement it.

Design:

- The design of BeerClan is inspired by (Zhe & Naftaly H.). We wanted to make the model as decentralized as possible and most of the design decisions were made keeping this thought in mind.
- The only centralized component of this model is the Law Server. The law server also makes sure that the names of the community members using that law are unique, this saved us from using a centralized naming server.
- We also made use of the gossip protocol to implement some functionalities (approvals to join and removals from the community). Gossip protocol has a known drawback that it might miss some targets in some cases but on the other hand, it makes system more decentralized.
- We make use of control state to store all the attributes required for a profile to function as a part of the community. This implies that we are not required to use a database. However, we believe that if needed, a personal database for each member can be easily included in the design.
- The communication between members are PUSH messages, meaning, the sender sends a message and the receiver receives it.
- At-least three founders are required to start a community and the founders are immortal, meaning they can't be removed.
- New members can be added by invitation only. One of the existing members has to send and invitation and then depending on the approvals from other members, a new member can be added.
- Members of the community can have interests. Interests are stored as a part of control state. This enables ***narrow casting.***

The following figure represents the anatomy of BeerClan.

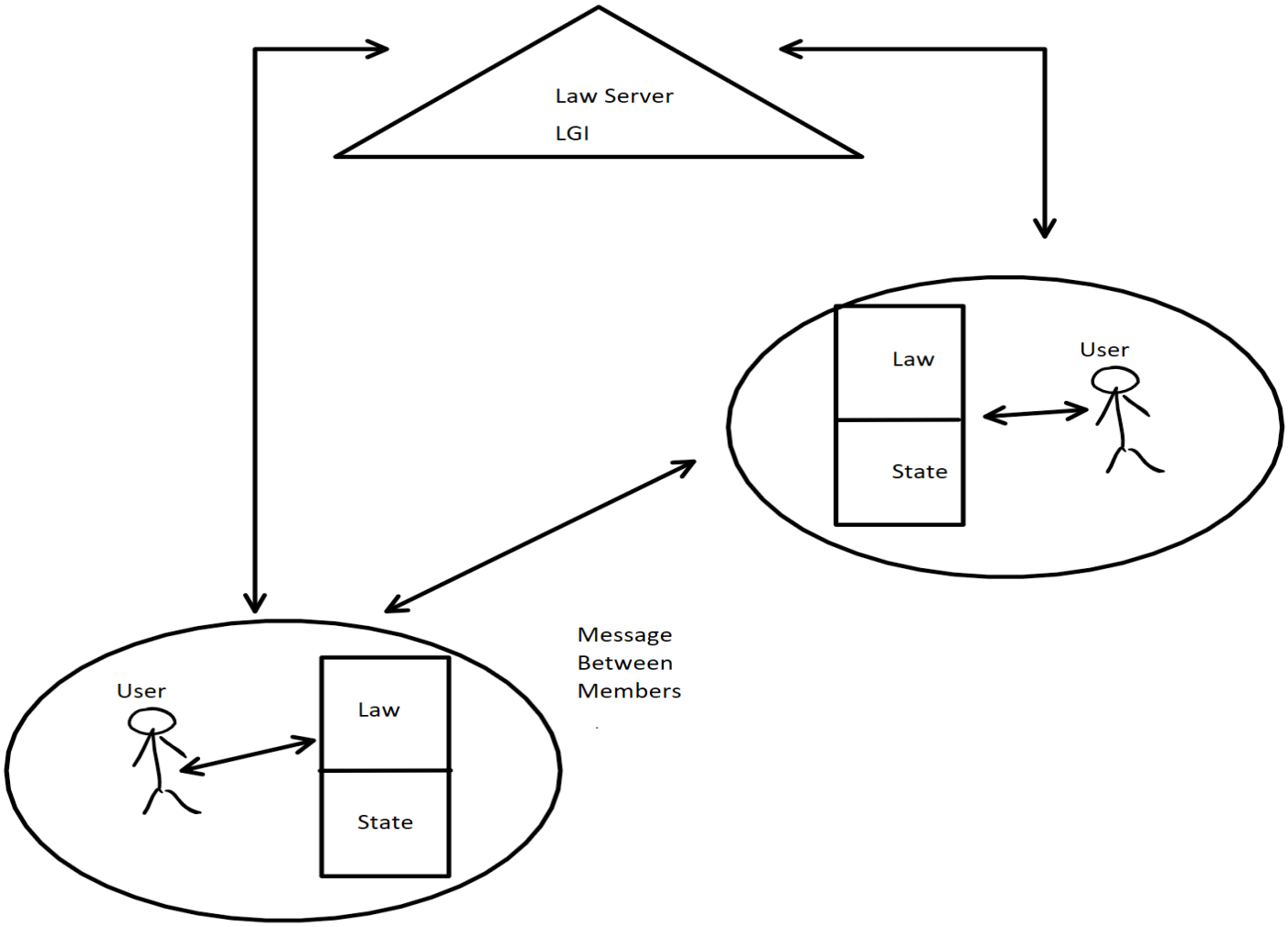

Figure 1. Anatomy of BeerClan

Gossip protocol: Gossip is implemented by remembering messages. This is done by storing a unique key for each message as a control state. This allows to ignore re-gossiping and gracefully terminates the gossip protocol. However, the application has certain limitations due to gossip:

- One cannot re-add a member after removing them, due to the persistence of gossip history.

- Another issue occurs while narrow-casting. One cannot search for the same interest again in the community.

Basic Capabilities of BeerClan and their implementation:

- **Membership control:**
  Adding a new member - A new member can only be added by invitation. This forces membership control in the community. Once a person (M) receives an invitation, he replies to the sender (S) of the invitation with a join-request. S then gossips this join-request to all its followers, which in turn gossip it to their own followers. Members may decide whether they want to approve or ignore the join-request. If a member decides to approve a join-request, it replies to the person (M) with his approval. After receiving three such approvals, the person becomes a member of this community. The membership information and approval counts are part of control state of the member.

  {"type": "invite"}

  Removing a member – An existing member can be removed from the community by disapprovals. Any existing member (S) who wants to remove another existing member (M) starts a remove-request gossip with its followers, who in turn forward the gossip to their followers. If three members decide to respond to the gossip by sending their disapproval to M, then M is removed from the community. Disapproval count is part of the member control state.

  {"type": "remove"}

- **Following:** An existing member (S) might decide to follow another existing member (M). In this case, S sends a follow request M and if M decides to allow S to follow, the S is added to M's follow list. This implies all the members that are following M are maintained in M's control state. Similarly, S might decide to unfollow M at any time which causes removal of S from M's control state.

  {"type": "addFollower"}

  {"type": "removeFollower"}

- **Posts:** If a member S posts something, then that post is pushed to all its followers; a list that is part of M's control state.

{"type": "post"}

- **Friends:** Since follow is a one-way interaction, we decided to include another relationship called friends. If an existing member (S) wants to be friends with another existing member (M), the S sends a friend-request to M. If M is also willing to be friends with S, then upon receiving the friend-request, S is added to M's friend-list. Friend-list again is an attribute maintained in the control state of each member. After adding S to the friend-list, M replies to S with an acknowledgement. Upon receiving the acknowledgement, S adds M to its friend list. At this point, M and S are mutual friends. To keep track of sent friend requests, we use a ping-pong type mechanism using a control state attribute. Any member S can decide to remove a friend M at any time. After removing M from its friend-list (control state attribute), S sends a remove-friend directive to M. This makes M remove S from its friend-list.

{"type": "addFriend"}

{"type": "removeFriend"}

- **Personal Messages:** The whole point of having the concept of friendship is to support an extra mode of communication in the form of personal messaging. An existing member S can send a personal message to M, provided M and S are friends. Following a member is not enough to send him/her a personal message.

{"type": "pm"}

- **Narrow Casting:** Every member can have some interests associated with the profile. The concept of interests allows us to narrow cast messages based on interests. If a person S wants to know someone with shared interests, he/she can start a gossip with that interest. If another person M shares that interest, then S will receive a message from M that would allow S to know M's identity. However, S cannot talk to M at this point. A friend request has to be sent first if S wants to talk to M.

{"type": "addInterest", "topic": "xxx"}

{"type": "narrowCast", "topic": "xxx"}

Code: Following is the link to an online git repository that contains the law file (bc.law) for BeerClan.
https://github.com/alok-sin/lgi-beerClan

Future Work

- Right now, BeerClan cannot add a member again immediately after removing it from the community. This is due to persistence of gossip history in the controllers. In the future we would want to use obligations to remove the history after a certain time period so that future operations are not affected.
- Overcome the limitations of the gossip protocol- We want BeerClan to be as much distributed as possible and hence we used gossip protocol which comes with its own shortcomings. There is a possibility that it might not reach all its targets in some cases. In the future, we would want to come up with a way to overcome this.
- Develop a user interface: BeerClan right now works as a command line application. In future, we would like to develop it into a web application.

References

Cong, C. (n.d.). Writing LGI Laws in CoffeeScript. Retrieved from research.cs.rutgers.edu/~cc1156/dropbox/manual.pdf

http://www.moses.rutgers.edu/. (n.d.). Retrieved from moses.

Minsky, N. (2005). *Law Goverened Interaction (LGI): A Distributed Coordination and Control Mechanism.*

Zhe, W., & Naftaly H., M. (n.d.). A Novel, Privacy Preserving, Architecture for Online Social Networks. *EAI Endorsed Transactions*.

----------The End----------