Adda: Peer to Peer Group Chat

Rahul Chatterjee, Brandon Davis, Saikat Raphael Gomes

Introduction

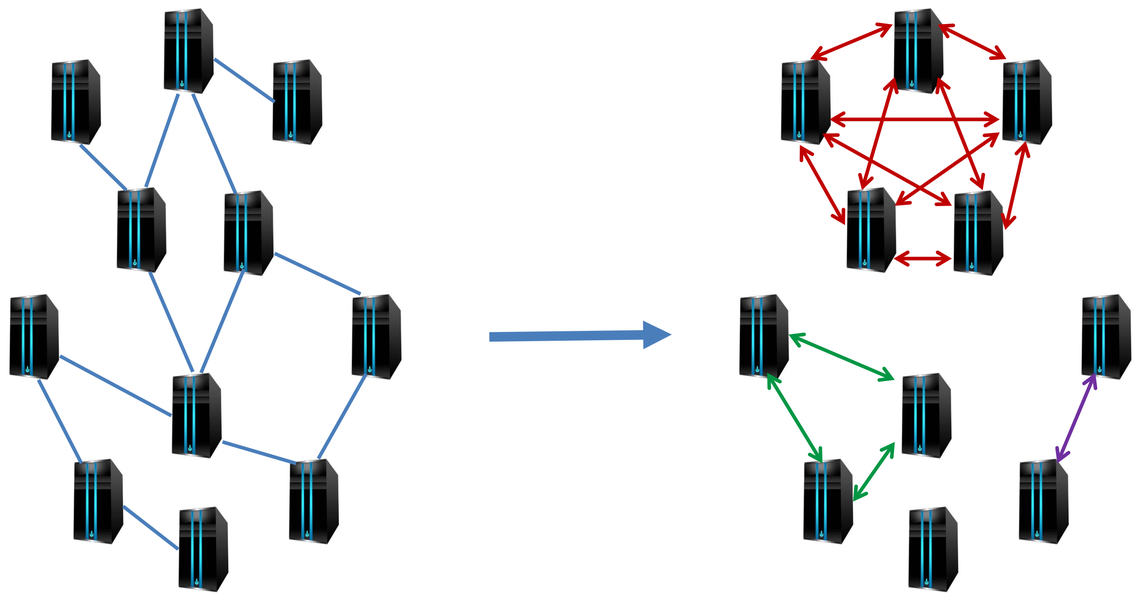
Adda is a secure, peer to peer replacement for messaging systems such as Google Chat, Facebook Chat, WhatsApp, GroupMe, etc. It is built on top of Treffen, our redesign of the Kademlia DHT used for finding peers.

Goals

- **Peer to Peer** – Data/control decisions flow between peers with no third party interaction.
- **Decentralized** – No single point of failure for routing or group chat.
- **Quickly Consistent Transcripts** – All users have same view of chat proceedings, and client view becomes stable with low delay.
- **Confidentiality** – Message contents only recoverable by those who are group members at the time of its sending.

Non-Goals

- **Anonymity** – Message contents are secret, but group existence and membership is not.



Treffen Overlay

Adda Overlay

Locating Peers

Kademlia

- Peer to peer distributed hash table similar to Chord or Pastry
- Nodes get random ID in 160-bit space
- XOR distance metric is used between IDs
- Each node maintains a routing table of K-Buckets, which contain up to K nodes with the same ID prefix (Circles in Figure correspond to K-Bucket ranges)
- So nodes may know about many nodes “close by” and fewer nodes further away

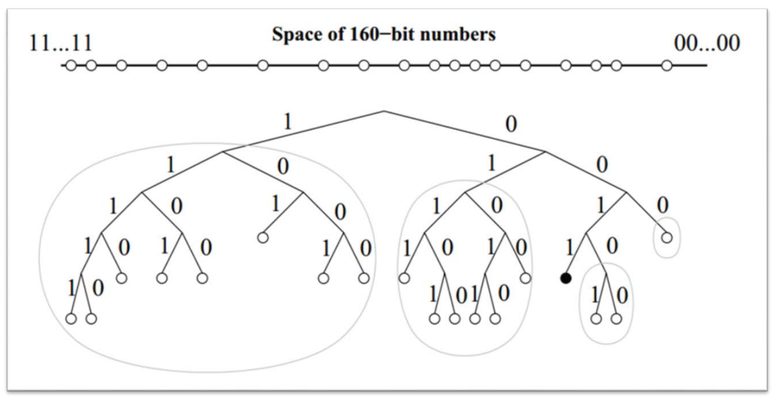


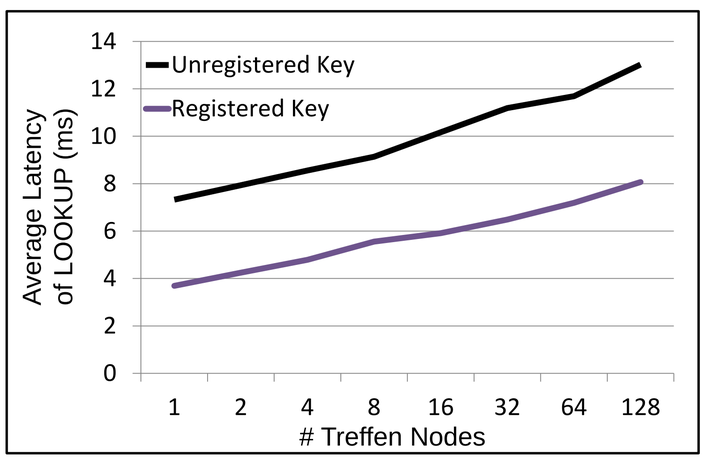
Figure from [1]

- PUT(key, value): store pair at K nearest nodes to ID hash(key)
- GET(key): ask nodes closer and closer to hash(key) for value
- **Pro** – Logarithmic in number of nodes
- **Con** – Membership changes can lead to stale or missing data

Treffen fixes Stale Data

- Node has no ID at startup
- REGISTER(key, <host, port>): Node will join network acting with ID=hash(key)
- Multiple registrations exist at each node, and registrations expire after some time
- LOOKUP(key): Node will search for hash(key) and retrieve saved <host, port>
- Registrar must be found exactly, not at “nearest K”, so no consistency problems

Registration of 500 Peers	
REGISTER One by One	0% LOOKUP Failure
REGISTER in Parallel	48% LOOKUP Failure
Latency of First	12.17 ms
Latency of Last	20.75 ms



The Chat Service

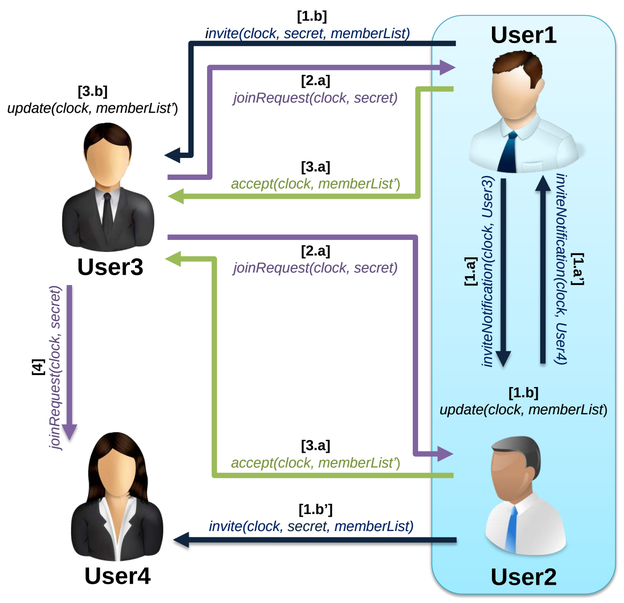
Overview

- $\binom{n}{2}$ TLS connections among group members
- Anyone can invite anyone to a chat, invitee can chose to decline
- Anytime anyone can leave (might not be gracefully) the group
- Anyone in the group can kick anyone out!
- Anyone can request to join another group

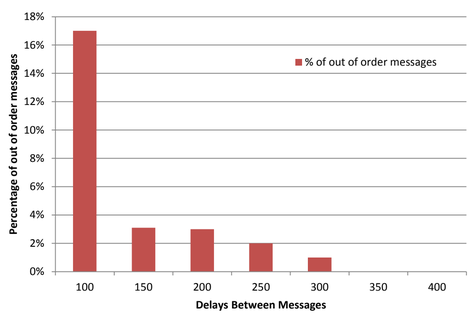
Message Ordering

- Goal: Everyone in the group shares same message transcripts in the absence of membership change.
- Using Lamport’s logical clock
- Everyone maintains his view of the ‘global clock’ and increment accordingly with every network event.
- Everyone sends his clock along with any message it sends.
- Messages are ordered based on the clock. Ties are broken by the order of public key.

Membership Changes



Message Reordering



Protocol	TLS 1.2
Key Exchange	RSA
Data Integrity	HMAC-SHA1
Encryption	AES-CBC

Encryption

Goal:

- Secure
- Authenticates users
- Fast to setup and encrypt

Why not multiparty Diffi-Helman Key Exchange ?

1. Too much setup overhead on membership change.
2. Benefits only Large groups, but large group has more expected number of membership change.

References

[1] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In Peer-to-Peer Systems, pages 53–65. Springer, 2002.
[2] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. ACM SIGCOMM Computer Communication Review, 31(4):149–160, 2001.
[3] “WikiPedia: Transport Layer Security,” http://en.wikipedia.org/wiki/Transport_Layer_Security
[4] L. Lamport. Time, clocks, and the ordering of events in a distributed system. Communications of the ACM 21, 7, 558–565