Peerchat: a Distributed, P2P Communication Network based on Kademlia

Forrest Pieper
Will Drevo
Colin Taylor

May 4th, 2014

1 Motivation

What we today know as the "internet" was started as a US DARPA military project, a network of nodes distributed geographically across the US in order to ensure fault tolerance in the case of a nuclear attack [**?**]. Today, ironically, many feel the internet has become too centralized.

2 Introduction

Peerchat is a distributed, P2P chat system based on the Kademlia DHT [1] system which adapts Kamdelia to support persistence of users over different IP addresses using the same disk, and a distributed user directory with no central server.

To guarantee no central dependencies, Peerchat requires that nodes bootstrap with an explicit IP address of a peer in the Peerchat network in order to be introduced into the routing table and is parameterized by the standard k and α Kademlia parameters.

We implemented two main entities for each Peerchat node: *User* and *Node*. This separation ensures that the DHT functionality is separated from the chat protocol.

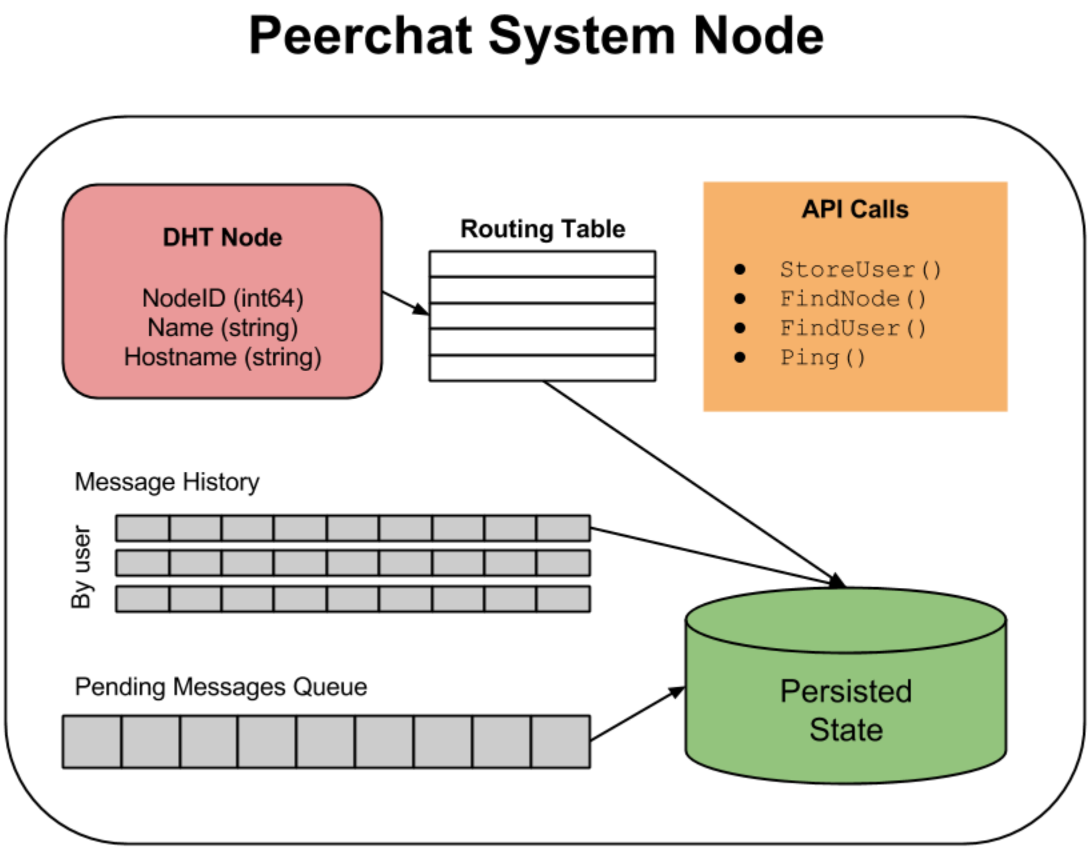
3 Background and Related Work

A very similar effort is BitTorrent Chat [**?**], which seeks to leverage the BitTorrent network to also also for personal, anonymous, decentralized communication. The effort has generated excitement due to the already massive user base of BitTorrent, though to date has not been released, nor is planned to be released as open-source.

We believe that a truly distributed chat system must be open-source in order to gain acceptance and ensure rigorous third-party audits. Though Peerchat in its current configuration is not secure, we feel that both a starting point and a commitment to open-source is important in a P2P chat client.

4 System

Below is a diagram of a single Peerchat node.



A Peerchat user consists of a number of different routines all working together. A short description of their function is below:

Separate threads (goroutines):

1. *Message Queuing*
   The message queuing routine is what acts when a user submits a message. The user's node sends the message to a queue where it is stored until the Message Sender can deal with the message. Additionally, messages that are being forwarded on in the case of offline messaging are also "piggybacked" into other user's pending messages queue, and await being forwarded to other nodes.

2. *Message Sender*
   The Message Sending goroutine iterates through the list of recipients with messages to be sent in the User?s queues. Next, using the DHT, the Message Sender attempts to locate and ping the recipient peer. If the peer is online, the message is sent directly through TCP using an RPC call. If the peer is offline, the Message is sent to the K closest neighbors in the DHT by the XOR metric to increase the chances of message delivery when the recipient happens to come online.

3. *Connection Acceptor*
   The connection acceptor is the Peerchat routine that accepts RPC requests and spins off new goroutines to service them.

4. *Periodic Backup*
   As well as backing up the system after each received message, Peerchat also backs up it?s message history, routing table, received and seen messages maps, and connection information every 30 seconds. This covers for unpopular nodes who are more often message ?forwarders? than receivers while avoiding spurious serializations to disk.

4.1 Persistence

We persist nodes' state by periodically serializing a user's routing table and messaging log to disk.

4.2 Offline Usage

Peerchat assumes online usage, but once a message has failed to send, falls back onto a K closest neighbors forwarding strategy.

Upon failure, a node also sends the message to the K nearest nodes around the node, which add the message to their background pending message queues, and attempt in their normal fashion to send them along, keeping the to/from message headers intact. This piggybacked forwarding strategy increases the chances of avoiding a situation in which both nodes continue to miss each other being online, and keeps the conversation going.

5 Demonstration

In addition to a battery of unit tests and simulations, we also present an analysis of the Peerchat's efficiency and and performance under different parameterizations of the system.

5.1 User Registration

colin

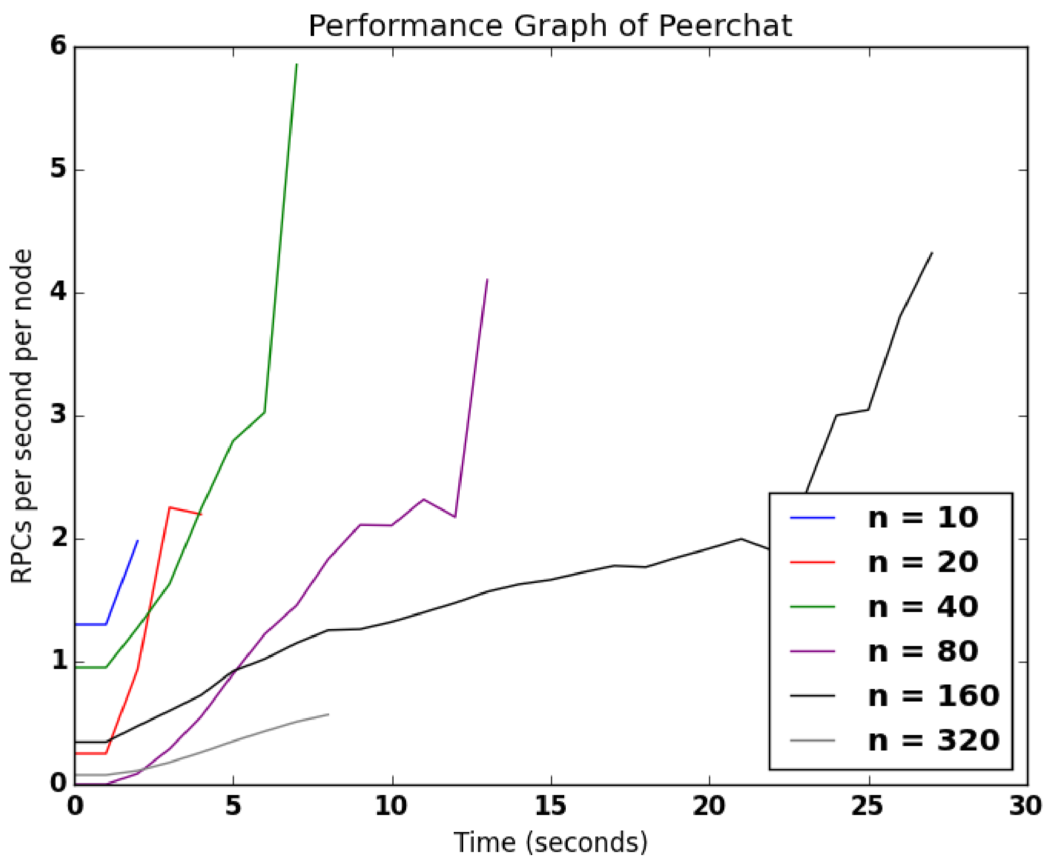
5.2 User Login

colin

5.3 Performance Testing

We tested the Peerchat system under a exponentially increasing size of the network n hosted on a single machine.

For this benchmark, we used our *TestManyMoreRegistrations* test, which simulates a network of nodes undergoing a tumultuous increase of registrations and users entering the network. We kept $k = 10$ and $\alpha = 5$ constant, and varied only the size of the Peerchat network.

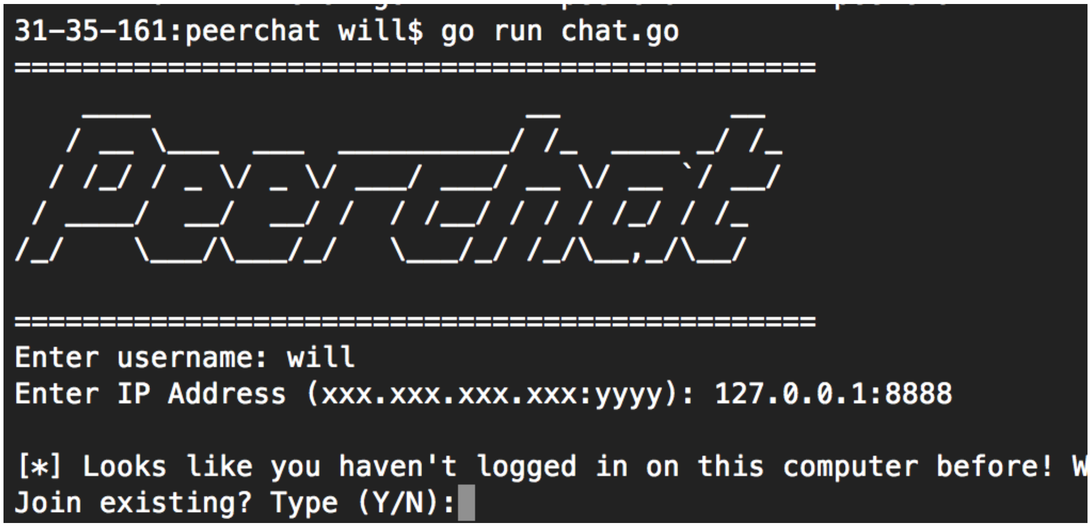


We see a clear correlation between n and the length of the test. Most interestingly, the rate of increase of congestion and throughput of the network *decrease* as n increases.

Unfortunately, the test of 320 nodes was cut short when our 4 core desktop could not handle the number of open files, but as you can see the slope of increase of the number of RPCs per second per node is actually the smallest of any of the tests. As the length of the tests more or less increases linearly even with the doubling values of n, we attribute the lower slope to the performance of the system rather than the limitations of the desktop.

5.4 Chat User Interface

We considered the terminal UI secondary to the DHT implementation, as it was only a necessity for testing, but have made a usable interface for terminal users on Unix systems.



Upon running the Peerchat script, the user is prompted to enter their username. If that username has not been logged onto the system before (if the gob backup file was not found in the Peerchat data directory), then the user is prompted to start a new network or join an existing one.

6 Future Work

Peerchat has many hole that need to be filled before it becomes widely used or trusted.

Security UI

7 Conclusion

Peerchat is the best, blah, blah.

References

[1] Maymounkov, Petar and David Mazieres "Kademlia: A Peer-to-peer Information System Based on the XOR Metric" *Peer-to-Peer Systems. Springer Berlin Heidelberg*, 2002. 53-65