YakStack: A Decentralized Yik Yak Built on the Blockchain

Margaret Li
Collaborator: Sarah Pan
Advisor: Michael J. Freedman
Fall 2017 Independent Work

Abstract

The blockchain is a revolutionary technology originally devised as the infrastructure of Bitcoin. Essentially a distributed database of information across a network of computers, the blockchain boasts three key features—decentralization, individual ownership of data, and tighter security—which make it relevant to all fields involving the storage and sharing of data. One emerging application of blockchain technology is in social networks, which transport user information across numerous devices and servers. Networks currently rely on a trust-based model, in which an intermediary stores and manages user data. However, the vulnerabilities of these intermediaries could jeopardize the security of the whole network and the safety of its users. Built on top of the blockchain, social networks can eliminate this central authority and increase security and transparency. This paper details our project to construct YakStack, a blockchain-based version of the defunct anonymous platform Yik Yak.

1 Introduction

Social networks currently follow an increasingly outdated architecture in which a trusted middleman, in the form of remote servers, stores and manages user data. Unfortunately, the middleman is a prime target for hackers, who frequently breach security barriers and steal data from millions of users.[1] This brings up the question of how social networks can be redesigned to eliminate intermediaries and thus the security hazards associated with them.

1.1 What is Yik Yak?

Yik Yak was a short-lived, anonymous micro-blogging application launched in 2013. The app contained a simple but well-liked interface: anonymous posts, upvote and downvote, comments, and a location-based list of trending yaks. Users did not have to create accounts and were instead assigned random logos and IDs, thus masking everyone's identity from each other. For many users, Yik Yak served as a haven for comic relief and for uniting a community over common struggles; however, a few users abused their anonymity and posted threats, putting Yik Yak under heavy scrutiny. Yik Yak rebranded itself as a non-anonymous group messaging platform, causing it to lose popularity before shutting down in 2017.[2]

Like other social networks, Yik Yak operated on top of a centralized model that rendered users susceptible to attacks on the server. Yik Yak's selling point was that posts were public but not publicly traceable to the poster, which encouraged users to post without any inhibitions. Anyone who hacked into Yik Yak's central system would gain access to not only records of who posted what, but also IP addresses, GPS locations, and phone numbers, which would endanger the confidentiality, reputation, and safety of the users.[3]

1.2 The Blockchain

The blockchain is a distributed ledger that records almost any form of information (e.g. transactions, smart contracts, user accounts, etc.). Originally built as the backbone of Bitcoin to address the flaws of trust-based financial transactions, the blockchain has expanded its use to nearly every field. Multiple computers form a peer-to-peer network that continuously updates the blockchain. When a transaction is executed, the network verifies it through a process called consensus. This process ensures transparency of information. The blockchain itself consists of a long chain of blocks, each of which stores verified transactions in chronological order. Each block is signed with the hash of the previous block. Thus, if

[1] Ali, Shea, Nelson, Freedman, *Blockstack: A New Internet for Decentralized Applications*

[2] Statt, *The Verge*

[3] Buntinx, *Bitcoinist.com*

someone were to alter information on the blockchain, the stored signatures would reveal that data has been corrupted; this means that all transactions on the blockchain are immutable and irreversible. Since all transactions on the blockchain are public, permanent, and verified, they are guaranteed security. Furthermore, each participant in the network stores and contributes to the blockchain, giving ownership of data to the participants and making it decentralized. These qualities make the blockchain a useful tool for redesigning centralized social networks. [4]

1.3 Goal

Our goal is to revive the early version of Yik Yak, and to build it in a decentralized manner on top of the blockchain. We chose Yik Yak because it was the first successful pioneer of anonymous content sharing platforms, and though it was embroiled in controversy over malicious users, the majority of content (at least from Princeton students) was innocuous, lighthearted, and worth the download. Decentralizing Yik Yak brings a couple of benefits:

- Individual ownership of data: the blockchain stores each user's information locally on his/her device, giving the user ownership of his/her posts. This creates a network of all devices using Yik Yak that work together constantly to update user information and contribute new posts.

- Security: because information is not stored in a single location, there no longer exists a central system that a hacker can target, and users can be guaranteed confidentiality of identity.

- Transparency: centralized social networks lack transparency when it comes to how they exploit user data. Since information is now stored across all devices on the network, no single authority exists to control this data. In fact, everyone on the network can view other users' posts without seeing the users' identities, which is a perfect set-up for Yik Yak.

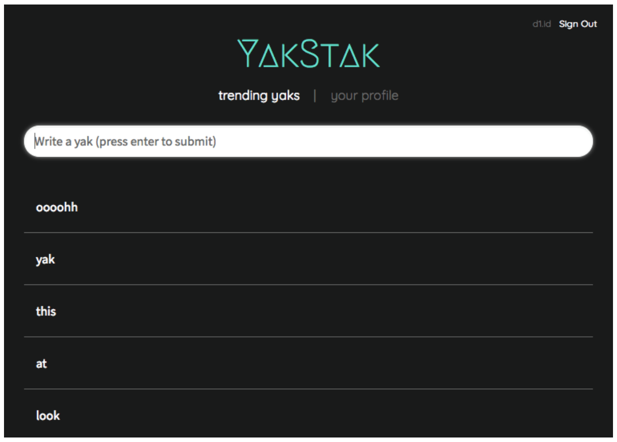
1.4 Related Work

There are a number of spinoffs that emulate certain aspects of Yik Yak, but none share quite the same features as the original. Hive (released by former Yik Yak employees) and Islands are chatrooms for college students to discuss classes and meet up, and they require usernames.[5] Whisper is an anonymous confessional platform that organizes content based on topics, rather than by location as Yik Yak did. Sarahah is an anonymous feedback app that encourages users to evaluate each other.

Decentralized social networks have been released, but they all require public user profiles and special tokens, making it harder for the average user to join. Steemit operates like Twitter but also compensates contributors of popular posts with a Steem cryptocurrency, and requires users to create wallets. Synereo, SocialX, MaskNetwork, and Yours.Network all follow a similar model that reward content contribution with tokens.

2 Implementation

In designing YakStack, we aimed for a clean, easy-to-use interface. The app consists of a "Your Profile" page, on which a user can post a yak or view his/her past yaks, and a "Trending Yaks" page, on which a user can view and vote on other people's yaks.



oooohh

yak

this

at

look

Figure 1: YakStack user profile page

YakStack is built on top of Blockstack, a blockchain-based browser created by Princeton alums. On Blockstack, users can register IDs that are stored on the blockchain and choose where their yaks are stored locally. We coded in Vue.js and Bootstrap, the same frameworks that Blockstack's examples use. We modeled some of YakStack after Blockstack's Todo app, a single-read application that stores a user's todo items.[6]

Centralized Yik Yak relied on servers scattered across different geographic locations. When a user posted a yak, the app would retrieve his/her GPS coordinates and send the yak to the server covering that location. If a user opened the app, the server in his/her region would send its yaks to the user's device. By contrast, our implementation is server-free and relies instead on the users' local computers, due to the architecture of Blockstack that I will detail below.

[4]Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*

[5]Newton, *The Verge*

[6]https://github.com/blockstack/blockstack-todos

2.1 Blockstack

Blockstack is a decentralized internet built on top of the blockchain that provides identity, discovery, and storage services. Users register human-readable IDs that are stored on the underlying blockchain, which enables discovery of other users on the network. Blockstack IDs are secure because of the immutability and transparency guaranteed by the blockchain. With an ID, a user can sign in to a variety of applications available on the Blockstack internet, such as YakStack.[7]

On centralized sites, user data is pushed to remote cloud servers, leaving the users' local devices with little to process. This places a burden on the servers to handle all the complexity efficiently and securely. Gaia is Blockstack's decentralized storage system that reverses this situation and keeps user data local without requiring users to run their own servers. In this system, the user selects a cloud storage provider where his/her data will live, such as Dropbox or Google Drive. Gaia then encrypts data prior to storing it in the cloud provider, ensuring that the provider (and anyone who hacks it) has zero visibility of the data. With this model, users can use their pre-existing storage services for Blockstack without having to trust them. Furthermore, unlike with Ethereum and other blockchains, data will not have to be stored on the blockchain, ensuring that the blockchain can scale.[8]

We used the following components of Blockstack to produce YakStack:

- Blockstack Core: reference implementation of Blockstack protocol
- Blockstack Browser: tool to run Blockstack internet using a local instance of Core
- Blockstack.js: library for profiles, authentication, and storage[9]

The most important component of YakStack is multi-reader storage, which is the ability to aggregate data from multiple users. This was the main challenge of our project, because data is confined to each user's device and must somehow be combined with other users' data without a central storage in place. The following paragraphs detail how a user would navigate YakStack, and how the app works with Gaia to enable multi-reading. Figure 2 maps YakStack's components.

2.2 User Authentication

Users are required to register a Blockstack ID (e.g. name.id) prior to signing in to YakStack. Clicking YakStack's sign-in button invokes Blockstack's authentication process (top of fig. 2). This sets off two actions.

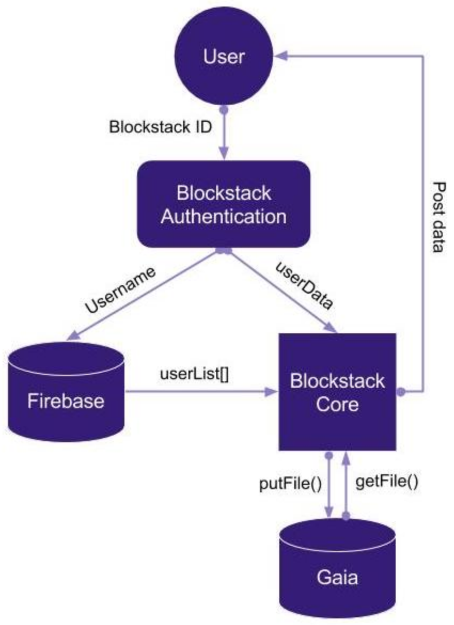


Figure 2: map of YakStack components

First, the user's data (ID, posts, votes) is sent to Blockstack Core (arrow from Authentication to Core in fig. 2). From there, Blockstack communicates with the user's local Gaia storage via Blockstack.js functions getFile() and putFile() to read and write files. Here, YakStack writes user posts to a JSON file that Gaia stores.

Second, the ID is recorded in Firebase (arrow from Blockstack to Firebase in fig. 2), which tracks the Blockstack users that have signed in to YakStack. This enables us to distinguish active users of YakStack from other Blockstack users, so that we only have to parse data from active users for multi-read storage.

2.3 Using Firebase for Multi-Reader Storage

Using a database to record IDs is not technically decentralized, but it serves as the minimum workaround to get multi-reader storage to work. Blockstack engineer Ken Liao told us that Blockstack does not yet support the ability to retrieve the users of a specific application, and recommended that we store the users in a database for the time being.

We decided to record only IDs, rather than additional data, for two main reasons. First, this keeps all remaining data local and secure in each user's Gaia storage, thus maximizing decentralization. Second, Firebase has no awareness of where or how the IDs are used; anyone viewing this database would simply see a list of ".id" strings. This means that Firebase has neither visibility

[7] Ali, Shea, Nelson, Freedman, *Blockstack: A New Internet for Decentralized Applications*

[8] Ali, Shea, Nelson, Freedman, *Blockstack: A New Internet for Decentralized Applications*

[9] http://blockstack.github.io/blockstack.js/index.html

nor access to any functions that would touch Gaia storage, so a hacker cannot exploit the IDs to retrieve data.

2.4 Single-Read with Gaia

When a user enters a post on the "User Profile" page, a yak instance is produced that contains the text and the votes it receives. This yak instance is then written to the user's Gaia storage via putFile(), illustrated by the arrow pointing from Blockstack to Gaia in fig. 2. Should a user upvote or downvote his/her own posts on this page, the vote number will automatically update in Gaia. The "User Profile" page also displays the user's past yaks and corresponding votes via getFile().

2.5 Multi-Read with Gaia

When a user goes to the "Trending Yaks" page, a process is set in motion to retrieve posts from other users. Blockstack retrieves the list of YakStack user IDs from Firebase (arrow from Firebase to Blockstack in fig. 2), then iterates through it and calls getFile() on each ID. Gaia returns a JSON for each user (arrow from Gaia to Blockstack), which the app parses and displays (arrow from Blockstack to User).

2.6 Multi-Write Caveat

Upvote/downvote currently has partial functionality. Although the vote buttons work on "Trending Yaks," meaning that the vote number updates on the screen, only the votes a user exercises on his/her own posts will actually be saved in Gaia. This means that a person's votes on other people's yaks will be reset when the page is reloaded. This issue stems from the parameters of putFile(). Most likely for security reasons, Gaia architecture currently allows users to read data from other users, but not write data to other users. As a result, unlike getFile(), putFile() does not accept an ID argument and automatically writes to the local storage of whoever is logged in, restricting it to single-write. Therefore, a user can only alter the votes of his/her own posts because he/she cannot sign in to other users' accounts. This also explains why a commenting feature would not work. The only solution we could brainstorm is a centralized storage mapping votes to posts (to comments), so that a single call to putFile() would update all votes properly; however, this contradicts our goal of decentralizing Yik Yak, so we chose not to implement it.

3 Evaluation

To register Blockstack IDs and test our app, we used the regtest environment, which consists of Chrome Incognito mode and Docker. In this mode, we could wire fictional Bitcoins to our account and use them to purchase Blockstack IDs, allowing us to test YakSatck with multiple users.

Time to Load Posts for One User (ms)						
# Posts	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Avg.
10	0.265	0.235	0.375	0.200	0.240	0.263
20	0.215	0.215	0.310	0.225	0.205	0.234
30	0.235	0.210	0.435	0.215	0.205	0.260
40	0.205	0.345	0.200	0.215	0.195	0.232
50	0.220	0.250	0.255	0.455	0.190	0.274

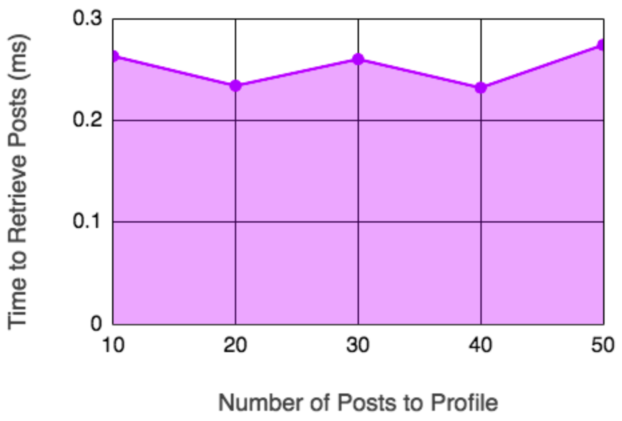


Figure 3: Latency of Single-Read Retrieval

We decided to examine the latency of data retrieval, since our app is constantly updating and fetching data from storage. We divided our evaluation into two tests:

1. How long does it take to retrieve N posts from one profile?

2. How long does it take to retrieve one post from each of N profiles?

In our code we inserted performance.now() at the start and endpoints of data retrieval, which returned two millisecond timestamps to us; calculating the difference between the two gave us the runtime.

3.1 Single-Read Latency

The first test times how quickly the "User Profile" page loads upon sign-in, and is effectively a measure of single-read latency. On a test user account, we created $N = \{10, 20, 30, 40, 50\}$ posts, then ran 5 trials for each N. The results are illustrated above in Figure 3.

The graph in Figure 3 suggests that single-user retrieval latency is constant time. This is corroborated by how our method for fetching data works. The app receives the user's JSON via getFile(), which takes constant time, and displays a parsed version of the JSON. From this test, we concluded that single-read is efficient and scalable for large numbers of posts.

Time to Load All Posts On First Retrieval (ms)						
# Users	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Avg.
10	3.665	3.260	4.225	3.785	3.675	3.722
20	4.091	3.899	3.913	4.218	3.978	4.020
30	4.145	4.789	4.683	4.597	5.088	4.6604
40	4.816	5.535	4.901	5.436	5.192	5.176
50	5.524	5.312	5.402	5.891	5.851	5.596

Time to Load All Posts On Subsequent Retrievals (ms)						
# Users	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Avg.
10	0.385	0.395	0.420	0.385	0.470	0.411
20	0.670	0.540	0.540	0.660	0.510	0.584
30	0.653	0.832	0.981	0.923	0.826	0.843
40	1.225	1.054	0.979	1.160	1.390	1.162
50	1.316	1.233	1.100	1.235	1.130	1.203

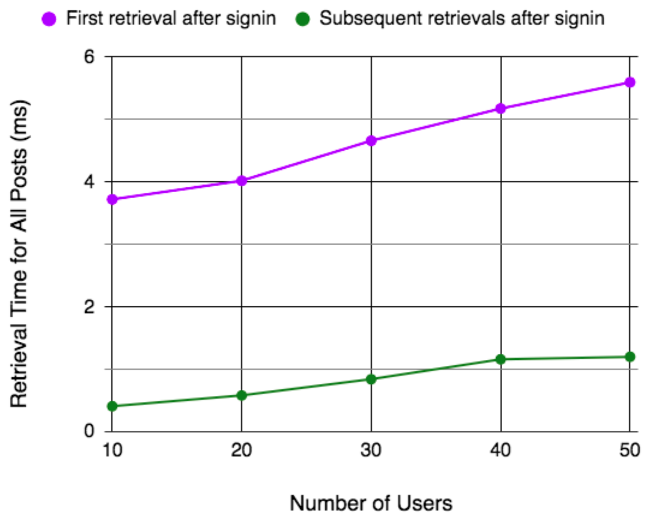


Figure 4: Latency of Multi-Read Retrieval

3.2 Multi-Read Latency

The second test times how quickly the "Trending Yaks" page loads, in two cases: 1) when the user clicks on this page immediately after sign-in, and 2) subsequent occurrences when the user clicks on this page again. We tested both cases because we expect a user to go back and forth between the two pages as more yaks are posted. We registered $N = \{10, 20, 30, 40, 50\}$ Blockstack IDs, posted one yak on each account, and loaded "Trending Yaks" from one of the accounts. Our results are displayed in Figure 4.

The graph suggests that latency in both cases runs in linear time with respect to the number of actively posting users. The latency immediately after sign-in takes a couple milliseconds longer than subsequent loads. This linear trend stems from the multi-read process, in which Blockstack traverses Firebase's list of users and runs constant-time getFile() on each user, thus being a linear time operation overall. We would normally run trials on a higher amount of users, such as 100, 200, and so on to verify this trend, but the processing time for registering new IDs forced us to cap the amount at 50.

Since multi-read latency increases with more active users, it could become a performance bottleneck as the user base grows larger. Overall, single-read is efficient, while multi-read could improve.

4 Future Work

We evaluated that YakStack has strong single-read performance for reads and writes from a user's account to his/her Gaia storage. However, the decentralized nature of BlockStack imposes restrictions on what its functions can achieve, so YakStack still misses some features that Yik Yak contained. As Blockstack adds new multi-read and multi-write capabilities, we plan to add more functionality to YakStack, so it becomes a full-fledged version of Yik Yak. Below, I will detail next steps for implementing the missing features.

4.1 Improving Multi-Read

Our multi-read currently requires a minimal database of user IDs. From our conversation with Ken, it seems that Blockstack will release a function to aggregate the IDs of users for each app. If so, we can transition from Firebase to that function, which will bring more decentralization.

4.2 Adding Multi-Write

As described in Implementation, getFile() and putFile() have different scopes. Whereas getFile() contains a parameter for user ID and thus enables multi-read, putFile() automatically writes to the logged in user's storage, which makes it single-write. This creates the issue where a user's votes on his/her own content will save, but votes across users will not. For the same reason, comments across users will not save, so we did not implement this feature.

Assuming Blockstack releases capabilities for multi-write in the future, e.g. adds more parameters to putFile(), we will implement the relevant functions to enable full voting and commenting on YakStack. With a functioning voting system in place, our “Trending Yaks” page will become more relevant, since posts will now be interactive with all users.

4.3 Censorship

A serious problem that has plagued anonymous platforms is abuse of freedom by users. Lack of traceability emboldens certain individuals to post threats and cyberbullying attacks, which can turn anonymous platforms into a dangerous environment. To mitigate this situation, we could add a feature where if a post is downvoted sufficiently by users, we remove it and temporarily blacklist the ID of the poster. We could also add a report button, such that if the number of users who report a certain post exceeds a threshold, we will again remove the post and blacklist the poster.

5 Conclusion

This paper describes YakStack, a decentralized Yik Yak built on the Blockstack blockchain with the goal of increasing security and user ownership of data. Centralized systems are trust-based and therefore leave users more susceptible to attacks. YakStack aims to solve this issue with a no-trust model, where data is stored locally whenever possible and any data recorded in Firebase cannot be exploited. This is especially important to an anonymous social network like Yik Yak, which guarantees confidentiality of identity to its users. The decentralized nature of the blockchain imposed some restrictions on how we could implement multi-read and multi-write capabilities, but as Blockstack continues to evolve and add new features, we hope to bring more functionality to YakStack and revive anonymous platforms.

References

Casey Newton. February 13, 2017. “Yik Yak Is Secretly Pivoting to Group Messaging.” *The Verge*. www.theverge.com/2017/2/13/14602798/yik-yak-pivot-hive-group-messaging-college

J.P. Buntinx. July 26, 2015. “Decentralized Messaging App Yik Yak Announces Anonymous Photo Sharing While Collecting Data.” *Bitcoinist.com*. https://bitcoinist.com/decentralized-messaging-app-yik-yak-announces-anonymous-photo-sharing-collecting-user-data/

Muneeb Ali, Ryan Shea, Jude Nelson, and Michael J. Freedman. October 12, 2017. “Blockstack: A New Internet for Decentralized Applications.” *Blockstack Technical Whitepaper*. Blockstack PBC.

Nick Statt. April 28, 2017. “Yik Yak, once valued at $400 million, shuts down and sells off engineers for $1 million.” *The Verge*. https://www.theverge.com/2017/4/28/15480052/yik-yak-shut-down-anonymous-messaging-app-square

Satoshi Nakamoto. 2008. *Bitcoin: A Peer-to-Peer Electronic Cash System*.