BIRB: Distributed Social Media Application over NDN-ICN

Saurab Dulal,
sdulal@memphis.edu
The University of Memphis

Abstract Name Data Networking(NDN)[1][11] – Information Centric Architecture (ICN)[12] is a emerging network architecture which is fundamentally different[14] than TCP/IP, the current networking architecture. There are several synchronization protocols such as ChronoSync[4] and PSync[8] developed for NDN. However, there exist a very few applications using these protocols and the core architecture of NDN itself. Specifically, there is no social media application using NDN, the main focus of this project.

This paper discusses about a distributed social media application called Birb – a twitter like application using NDN. The application demonstrates the usability of distributed communication over NDN using the above-mentioned synchronization protocol ChronoSync and PSync.

1 Introduction

Name Data Networking (NDN) is a data-centric internet architecture which uses the concept of naming a content rather than naming a host (e.g. 192.168.5.75) as done by the current IP based internet architecture. Similar to the client and server in TCP/IP, it uses the concept data producer and data consumer. The producer is the one who produces the data e.g. *The Newyork Times* , while the readers can be considered as the data consumers. Every single data released by the data producer in the network is a named content e.g. */memphis/operating system/project/birb.pdf*. To fetch this data, an interested consumer can send an interest packet with a name exactly matching the data name over the network. Interest packet is similar but not equivalent to a request in the TCP/IP network. Once the interest is received in the network by a node i.e. routers, it will check its cache - content store (CS) for the availability of the data corresponding to the interest received. If found in CS, the data is sent back to the consumer immediate on the reversed path followed by the interest packet. Otherwise, the interest packet is forwarded to the next node and so on. If the data is not found on any of intermediate nodes, interest will ultimately reach the data producer and hence the data will be served. The path to the data producer is stored in forwarding information base (FIB) of name-data forwarding daemon (NFD). More about NDN architecture can be found on the reference section [1].

Birb is a distributed desktop-based Twitter-like social media application developed for NDN architecture. It demonstrates the capability and the usability of NDN and several of its protocols developed for the communication and data transfer over NDN. Unlike most of the other social media platform developed for the TCP/IP networks, in terms of data publishing, Birb is purely a distributed application. All the data shared across the network are stored only on the local device of a user. Having said that, no intermediate servers are used to store the data content. More about the system design and the architecture of the application is explained in section 3. Before moving to a brief description of the application lets discuss on why choose NDN over TCP/IP to build a distributed application. There are numbers of notable benefits of ICN over TCP/IP which are as follows

a) Peer discovery is very complicated in TCP/IP networks: Theoretically, It is impossible to reach a device directly lying somewhere in next corner of the world or even the next neighbor if they don't share the same network. This is because almost all of the devices are under NAT in present-day IP net-

works and share some private IP address which can't be reached from outside. One possible solution is to this problem is NAT traversal - RFC 3957[17], RFC 3948[16]. But it is complicated to implement and requires an extra effort. I would have loved to discuss the NAT traversal but it is beyond the scope of this paper.

b) Availability of In-network caching and request/interest aggregation in NDN: As briefed in the first paragraph of introduction section, the CS of a node provides a heart to ICN network by caching the content previously fetched via it. To make it simpler let's assume a 1000 users from Memphis are trying to fetch a NewYork times articles from a server located at NewYork city. With TCP/IP, all 1000 request from these users will reach to the server (assuming no intermediate cache server are used) to get the article. But with NDN, since all 1000 user is trying to get the same article, all the request gets aggregated at the local node (say Comcast node), and only one request travels towards the server. It brings back the data and the local node distribute it to all the users.

c) End-to-End security in ICN: Security is one of the most important aspects of a social media application. Protecting the data, data centers, and the users' information are the biggest challenges of the internet today. Fundamentally, security is not a part of TCP/IP networks - layer 3. To achieve data security, several mechanisms are implemented on top of it but for the NDN, security is inbuilt. All the data packets are signed and encrypted making an end-to-end encryption much easier. Currently, NDN uses public key encryption by default but the architecture can support any encryption mechanism.
These are a few notable benefits of NDN over TCP/IP and there are several more[15]. Given these benefits and potential of NDN, it is much easier and cheaper – in term of packet exchange – to develop a secure social media application over it. Thus, making it my primary choice of building a distributed application attempting to solving some bigger problems like data security, and content distribution. The next paragraph discusses how the application works.

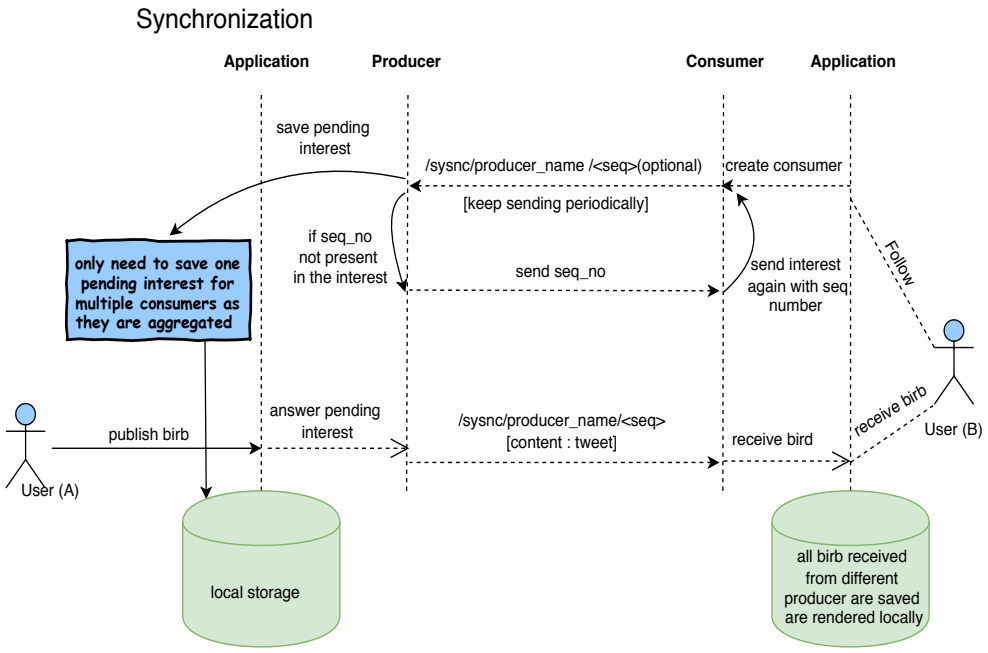


Figure 1: Figure: Sequence Diagram - Birb

An event sequence diagram of the application is shown in the figure 1. The figure shows two users A and B both already joined to the application. Joining to the application simply means installing the application on the local computer. This will create a local database for a user to store the data. Most importantly, it will let the world know under which prefix a user is publishing the data. For example, lets imagine User A is listening under a prefix "*/sync/birb/userA*". Listening under a prefix means if any request comes

for that particular name, upstream internet routers (nodes) will forward the request towards the one who is listening/serving under that prefix. This is a core of NDN hierarchical naming scheme and the way for another user to reach a particular user. We can visualize name prefix as an IP address of a data content but strictly not correlate them. The naming scheme is a beauty of NDN architecture, it doesn't require anything like NAT to create subnets because there can be several suffix under some particular prefix. Say, UofM is publishing data under name "*/memphis/uofm/*" prefix. Now, under this prefix any number of data name can be publish data such as "***/memphis/uofm/****distributed_sys/chapter1/transprancy/*" and so on.

The application(Birb) provides a search feature to find new friends. The search feature is achieved by storing a list of all the users registered to the application in an intermediate server. Using the search feature, User B can find and directly follow some other user, say User A in our case. Please note that, in the current design, any user can simply follow any other user and there is no involvement of followee in the following process. Once User B follows User A, it will send a synchronization (sync) interest to A. Interest in NDN is similar to the request in TCP/IP. Unlike TCP/IP - push-based system, NDN is a pull-based system. Here, data consumer will have to pull the data published at the producer's end. Upon receiving a sync interest, the application running on User A will either send a list of all the sequence numbers of the bird (technically data) it has published till date or will store the request in its pending table (PT) if no data has been published. The pending table is a simple database used for storing all the unserved pending interests and is used to serve all the pending interests received from the followers.

On the follower's side i.e. B, after receiving the list of the sequence number, B will start fetching the data corresponding to each sequence number one at a time. This is done by sending an interest */sysnc/producer_name (user A)/(seq_no)* to the producer (A). Every single piece of data received from the A is stored on the local database of B along with its sequence number. These stored data will be rendered by the application(Birb) running locally on the desktop of B. The interest from the data consumer i.e. B towards the producer i.e. A is periodic in nature. They are sent frequently in a certain interval. This regular interest is to synchronize between the follower and the followee whenever a new data is produced.

The remainder of this paper is organized as follows. Section 2 describes the related works. The design and architecture of the application are detailed on Section 3. The analysis on how the application incorporated the facts presented in the Introduction are presented on Section 4. Implementation challenges are presented in Section 5. Finally, Section 6 concludes the paper.

2 Related Works

Few applications are developed in the past similar to Birb that uses NDN as well as some other ICN architecture. SnapChat – a file sharing application that uses NDN to share files between the users. However, it has several limitations such as a) users needs to exchange public key via a barcode scanning b) it is not fit for remote data exchange as the testbed or any overlay network is not used. ChronoShare[10] - a distributed chat application that uses NDN to exchange message between users. It is a wonderful application facilitating data exchange but user availability is required to do so, also it doesn't support remote communication. ChronoChat[6] is another chat application similar to ChronoShare and has the same limitation as that of ChronoShare. nTorrent[5] – peer to peer file sharing application, similar to BitTorrent is another application developed over NDN. It creates an overlay of peers over the NDN network and supports file sharing.

In contrast to most of these applications, Birb mostly concerns with sharing status– similar to tweet in the twitter – and synchronizing other's status by maintaining a locally synchronized database between the follower and the followee. Also, none of the above-mentioned application (except nTorrent) facilitates remote communication between the users, neither they use the NDN testbed. Thus, Birb overcomes these challenges and shortcoming. It eases the remote communication by using the NDN testbed. It also facilitates distributed storage of the content on the user's local device whereby make the application pure peer-to-peer. The detail on the previously developed similar application can be found on the reference section.

3 System Design

A basic flow diagram of the application is shown in the figure 2. It shows a data exchange between two users A and B. The major component such as remote NFD, local database, local NFD, remote database server are presented on the figure and are described below.

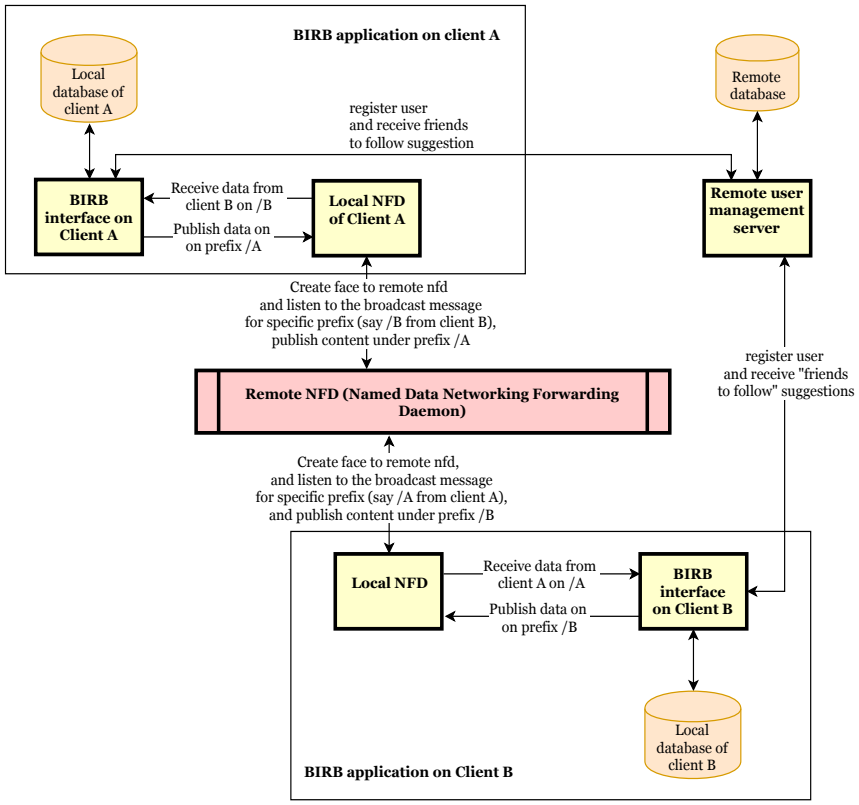


Figure 2: Flow diagram showing communication between two clients in Birb

3.1 User Registration

When the application is installed on a client machine, a minimalist user registration is required. By minimalist, it means just to provide a unique username during the registration process. Uniqueness is verified by querying the remote database containing a list of registered users. Once a user successfully registers to the application, following events takes place a) The username is stored in the remote database. For this, I have used a raspberry PI server http://75.65.50.70 and an object database to store the usernames b) A face is created from client's local NFD to remote NFD c) A database table is created on the client's machine to store all the data published and received by the client. These processes are illustrated on the 2. Below I have presented a pseudo-code of the user registration process.

3.2 Local Database

One of the major goals of the project was to support distributed data sharing and storage. To achieve this goal, a local object database is created on the client machine. This database stores all the data published by the client, its sequence number (incremented by one for each data published), and the published timestamp. Similarly, it will also store all the data received from the followee, as well as the followee's username. The database strictly only interact with the application. For a local user as well as for each followee two tables are created named <username>_birb and <username>_details. First table stores all the birb (data) published and received while the other table stores user details such as last sequence no of the data published, username,

Algorithm 1 User Registration Process

```
1: procedure RegisterUser
2:     username ← get username string
3:     if username exist in local env then return "application already install"
4:     bool = CheckIfUnique(username)
5:     if bool == True then
6:         return "Username already taken"
7:     OptainCertFromRemoteNFD() ← username.
8:     if Certification Failled then return "Certification process failed, try again later"
9:     else
10:        CreateLocalDatabase() ← username.
11:        RegisterPrefixToLocalNFD() ← username.
12:        StartListeningOnUserPrefix ← username.
13:    goto ApplicationHomePage.
```

no of followers and followee, and link to profile image. Followers can query details table to ask details about a user. All the information stored in these tables are rendered in the application home page.

3.3 Remote Server and Database

As the application promises to provide "follow suggestion" capabilities, a remote server and a database are deployed to perform this task. It runs a server-side PHP script to receive the registration request from all the new users and also provides an API to list all the users already registered to the system. A registration request is implemented using the POST method, and searching is implemented using the GET method. Before storing any username, it checks whether the username already exists or not. *CheckIfUnique* method in the above algorithm 1 is implement this server-side script. This is the only part of the application which is not distributed. Even though it is quite complicated to make this section distributed, but still can be improved in the future. Being out of the scope for this project, I have noted it for the future work.

3.4 Creating NFD face to remote NFD server

When a user register to the application Birb 3.1, a local forwarder(NFD) create a face (basically an interface) to the remote forwarder (NFD). This is shown in the figure 3 where Local NFD of A and B creates a face with remote NFD. Local NFD also notifies remote NFD about the prefix local application desires to listen on. Once the remote NFD validates local NFD request, it sends a certificate – a permit – to a local NFD to publish the data under desired name prefix, say /A or /B. This face helps to send the data to all the subscriber over the internet. When the remote forwarder receives the data from the local forwarder, it sends the data to all the subscriber of the data producer and vice-versa. There is one additional process called "Auto Prefix Propagation" involved to exchange the data. It is a process by which remote NFD identifies a legitimate user corresponding to a prefix. For example, if anything related prefix /A comes to remote NFD, it forwards that to User A.

Note that the remote forwarder doesn't store the data in the database rather it only acts as a bridge and helps to forward the data to the respective subscriber. It is similar to a router on the TCP/IP system.

3.5 Data Publishing

Data publishing is an easy process. A user can simply publish a data using the application interface, refer to Appendix section 5 to view the interface. The published data, sequence number, and the time stamps are also stored in the local database. Along with it, the latest sequence number is also updated in the details table.

3.6 Data Synchronization

Data synchronization (DS) facilitates an exchange of data between the follower and the follwee. It ensures that all the data published by the followee is received by the follower. DS is a crucial part of the application. Referring to figure 3, let us assume User B just joined the application. As explained in the above sections, joining will create a database for B in a local environment. B's details and all the data published by B is stored in the local tables i.e. <B>_birb and <B>_details table. Now using the search option, B followed A. It will first send synchronization (sysnc) interest to A. Upon receiving the sysnc interest, A will send the latest sequence number of the data it has published. If no data has been published by A, it will just send 0. B will receive the latest sequence number from A and will start fetching the birb (data) one by one, subsequently storing them in the local database. Once the fetching process completes, B will start sending periodic interest to A containing sequence number greater by 1 than previously received. This interest is intended to get the future data A will publish. Thus, B and A will always remain in synchronization. Please refer to 2 for the algorithm of data synchronization

One important point to note is that once the data received by the follower is stored on the local database, they are never fetched again. These data stored on the client database are displayed on the application UI.

Algorithm 2 Data Synchronization

```
1:  procedure DataSyncronization
2:      username ← get username string
3:      if FollowingNewUser then
4:          FollowUser() ← followee username string
5:          if FollowSuccessfull then
6:              latestSeqNumber = SendSyncInterst()
7:              while r == latestSeqNumber.length do          ▷ fetch all the data one by one
8:                  data ← SendInterest(DataName, SeqNumber)
9:                  StoreData ← data
10:                 UpdateSeqNumber()
11:                 UpdateDetailsTable()
12:         else
13:             "Failed to follow <username>"
14:     else
15:         SchedulePeriodicInterest()
16:     goto ApplicationHomePage.
```

3.7 Application Interface

Application interface implements all the functionality discussed above. The current design has three page, login page, index page, and profile page. Login page 3a is a minimal one and contains only one form field to supply a username. A new user can register to the application, or log in to the application using this page. Index page 3b renders all the data published and received by the user. It also contains a friend suggestion lists pulled from the remote application server/database. Additionally, it provides a form i.e. textarea to publish the data (birb), and a button to view the profile as well. Profile page 3c contains information fetched from the user details table. It renders the username, name, no of followers, no of following, and profile picture URL. Currently, the design is very simple and I hope to modify it in the near future.

4 Analysis

Application Birb can be analyzed in terms functionalities such as successful user registration process, peer discovery, data storage in the local device, face creation from local NFD to remote NFD, data encryption, and searching mechanism. The user registration process is done on a local computer. To provide the uniqueness in the username, before the registration, the remote application server is consulted. This process was tested

numerous times and is fully functional. Currently, the user data is stored on node.js default userData directory. Peer discovery is achieved with the help of remote NFD. Once the user register to the application, remote NFD issues a certificate to a new user. Without a certificate, no communication is possible in NDN. The certificate generally contains a public key signed by a trust anchor. In our case, remote NFD is the trust anchor. To be a trust anchor, remote NFD first obtains a certificate from the NDN testbed.
Now the face creates from local NFD to remote NFD can take place once the certification and user registration process completes. I used Memphis testbed node for testing the face creation process and was successful in doing so. The reason to use Memphis node is that the other NDN testbed nodes are managed by different operators and may require their permission to do the testing. As mentioned several times, data security is inbuilt in NDN. Every single data packet is signed before publishing. These data packets can only be decrypted by the intended user. Lastly, for the user searching, I used a raspberry PI at my home as a remote application server, and a MongoDB to store the usernames. The server is still up and can be accessed at http://75.65.50.70

5 Conclusion and Future Works

The growth of NDN-ICN and its architecture has open wide rooms for the development of secure data exchange application. The inbuilt security, content caching, and hierarchical naming schemes provide a good playground to develop the distributed application. With this paper, I proposed and developed a distributed application called Birb. Some of the key contributions of this work can be listed as a) used a local database to store content, b) NDN testbed certification for the end-to-end data security, c) prefix propagation for the retrieval and publishing of content, d) remote NFD for the face and routes creation and, e) remote application server to support user searching capabilities. It also demonstrated the capabilities of several protocols such as ChronoSync and Full Sync designed for NDN communication. Consequential, Birb is a good starting point for building a distributed social media application over NDN.

However, there are several limitations of this application. Use of remote application server undermines its distributed nature. The remote communication is still not fully achieved and requires some serious work, such as multithreading. The application interface is not well developed which can be very frustrating for a new user. Last but not least, the code is not robust enough for the immediate use, thus needs several modifications. I will continue to work on this project, and in the near future, I will try to come up with a fully functional distributed application overcoming the above-mentioned limitations.

References

[1] Named Data Networking by Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, kc claffy, Patrick Crowley, Christos Papadopoulos, Lan Wang, and Beichuan Zhang ACM SIGCOMM Computer Communication Review (CCR), July 2014.

[2] Networking Named Content by V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, R. L. Braynard CoNEXT 2009, Rome, December, 2009.

[3] Securing Network Content by D. Smetters, V. Jacobson PARC Tech Report, October 2009.

[4] Let's ChronoSync: Decentralized Dataset State Synchronization in Named Data Networking by Zhenkai Zhu and Alexander Afanasyev, in Proceedings of the 21st IEEE International Conference on Network Protocols (ICNP 2013), Goettingen, Germany, October 2013.

[5] nTorrent: Peer-to-peer File Sharing in Named Data Networking Spyridon Mastorakis, Alexander Afanasyev, Yingdi Yu, and Lixia Zhang. In proceedings of the 26th International Conference on Computer Communications and Networks (ICCCN), July 2017.

[6] https://github.com/named-data/ChronoChat

[7] https://github.com/named-data

[8] https://github.com/named-data/PSync

[9] https://github.com/named-data/ChronoSync

[10] https://github.com/named-data/ChronoShare

[11] http://named-data.net/

[12] Google Tech Talks: A New Way to look at Networking - Van Jackobson Aug 2006

[13] http://ndndemo.arl.wustl.edu/

[14] An Examination of Information-Centric Networking via Content-Centric Networking: An Indepth Comparison with the IP Architecture, Daniel Griffin, Tuffs University

[15] Trossen, D., Reed, M. J., Riihijärvi, J., Georgiades, M., Fotiou, N., & Xylomenos, G. (2015, June). Ip over icn-the better ip?. In Networks and Communications (EuCNC), 2015 European Conference on (pp. 413-417). IEEE.

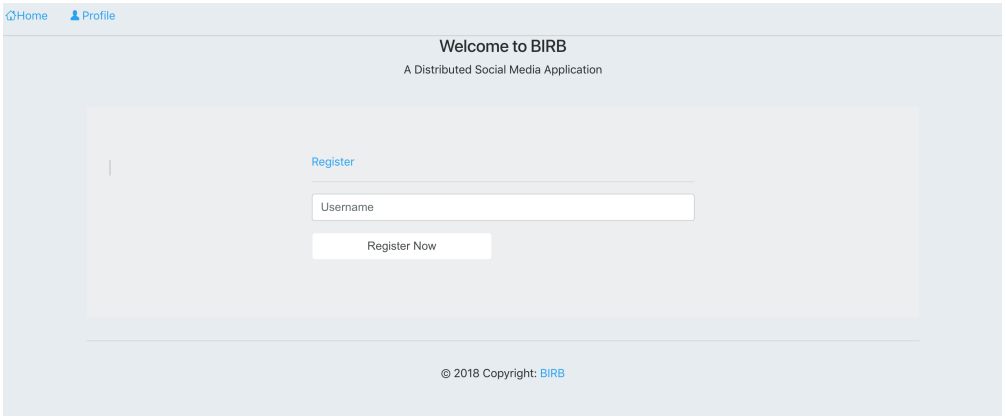
[16] A. Huttunen, F-Secure Corporation, B. Swander, Microsoft, V. Volpe, Cisco Systems, L. DiBurro, Nortel Networks, M. Stenberg, UDP Encapsulation of IPsec ESP Packets, January 2005

[17] Authentication, Authorization, and Accounting (AAA) Registration Keys for Mobile IPv4, C. Perkins, Nokia Research Center, P. Calhoun, Airespace, March 2005

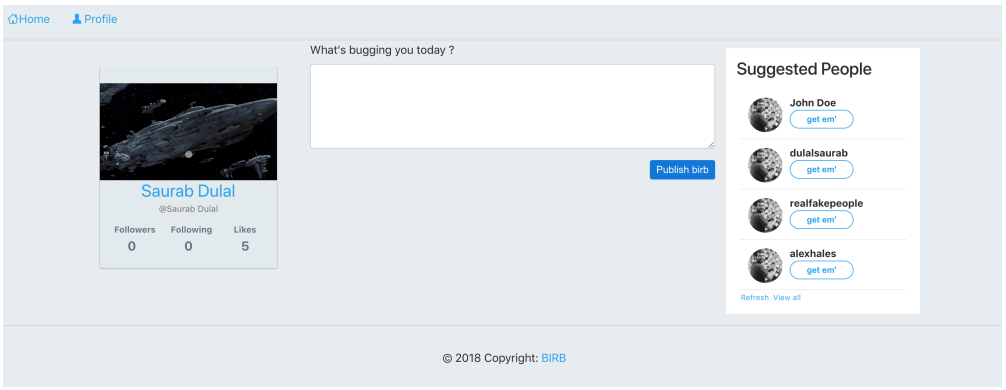
A Resource Materials

Source Code: https://github.com/dulalsaurab/BIRB
Installation Guide: https://github.com/dulalsaurab/BIRB/blob/master/README.md
Proposal: Birb: Project Proposal

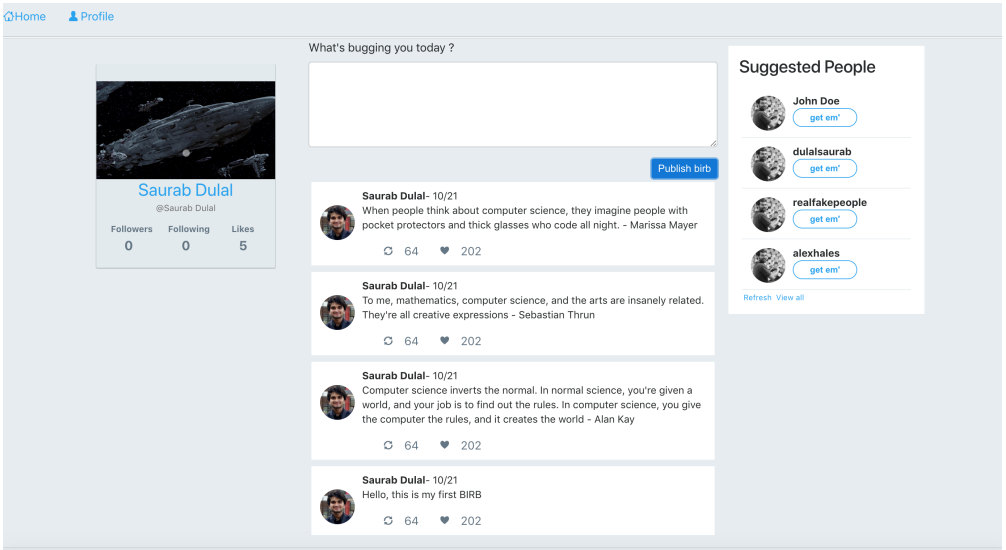
B Screen Shots



(a) Figure 1: Login Page



(b) Figure 2: Index Page



(c) Figure 3: Profile Page