**Quarterly Status Report**

**Report for period: Aug 01st, 2021 – Aug 31th, 2021**

Contract Number: #########

Title: #########

PI: **Sanjay Madria**

## Progress this month:

In the month, we made progress in the research of secure communication in DTN, content caching for DTN network, and the technique of the dependency tree and parse tree generation. The detail are as follows.

1. **Demo implementation**

This month, we have implemented the central authority. The front end of the central authority is implemented with React which enable the users to view and request for user and mission modification. The backend of the central authority is a Restful API built with java servlet and MySQL. The detail implementation are as follows

**Frontend of the central authority:**

The frontend contains the following components:

Welcome page has title “Central authority Demo”. It shows how many missions and how many users in the system. It also has entered place to “manage missions” and “manage users” page.

Manage missions page has title “Manage Missions”. It shows all the missions each of them has an enter place to manage that mission. It should have a float control bar that always float at the top or bottom or left or right that could add a new mission to the system. To add new mission, we need mission name (string) and mission capacity (int), start date, end date (chosen from calendar). When click each mission, we went to the single mission management page.

Single mission management page has title same as the mission’s name + “mission management”. It shows the mission’s name, capacity, start date, end data mission code and mission barcode (jpg).  It also shows a list of users already joined in the mission. It allows to change the mission’s name, mission capacity, start/end date and delete/add users. When adding users, it allows us to search the users with the user’s name, or search with user first name + last name.

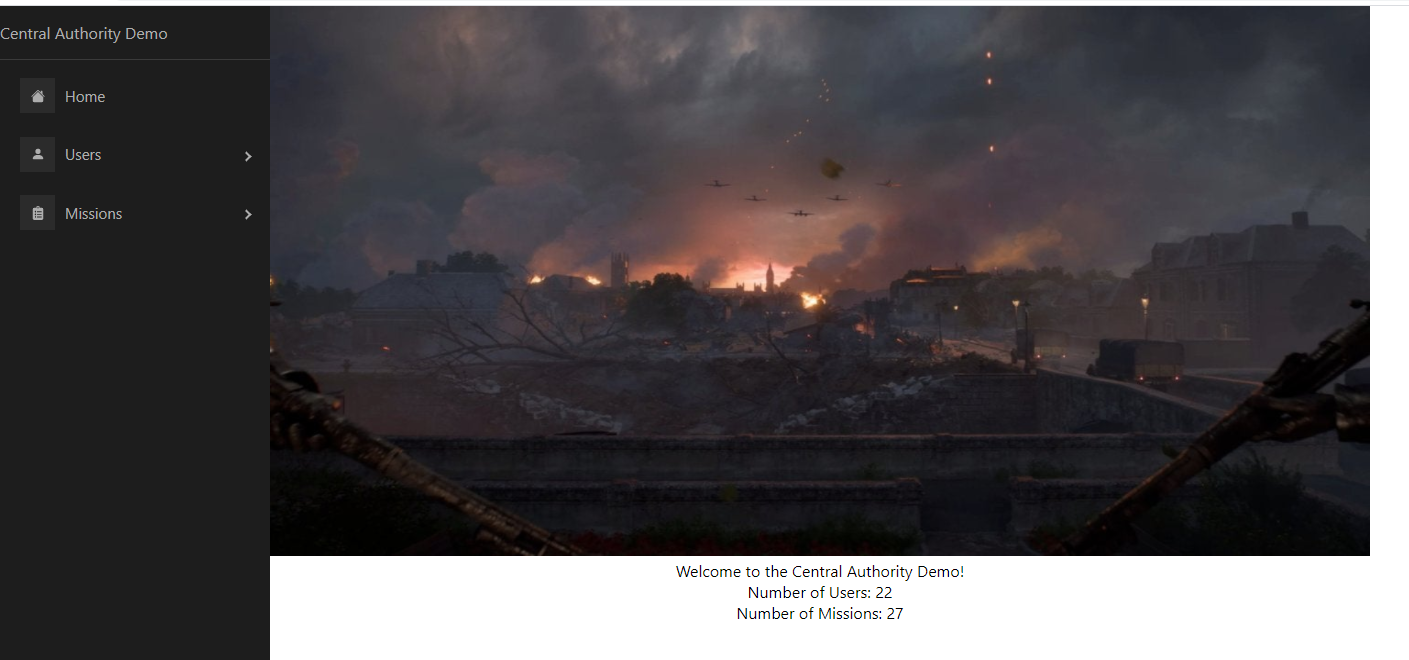


Figure 1. Home page shows the total number of missions and users

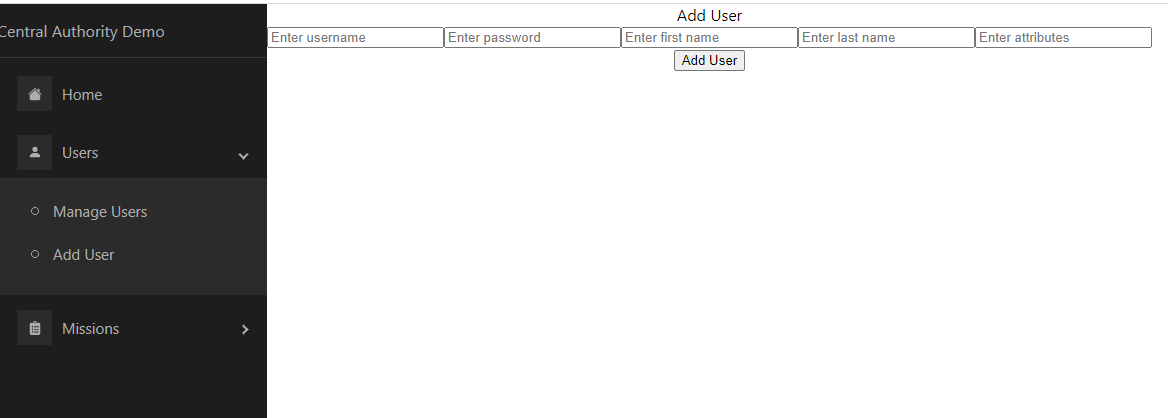


Figure 3. Page to add user to the system

As shown in the Figure above, the home page of our front end shows the title of the project and the total number of missions and users. The add user page shows a form to type in the new user’s name, password and expiration date.

For the Figure shown below, it is the page displays all the users in the system. By click any user in the link, we can modify that user by sending updating request to the backend of the central authority that make modification to that user.

Figure 4 shows the page that displays all the missions in the system. Click the link of any mission lead us to the mission setup page as shown in Figure 5.

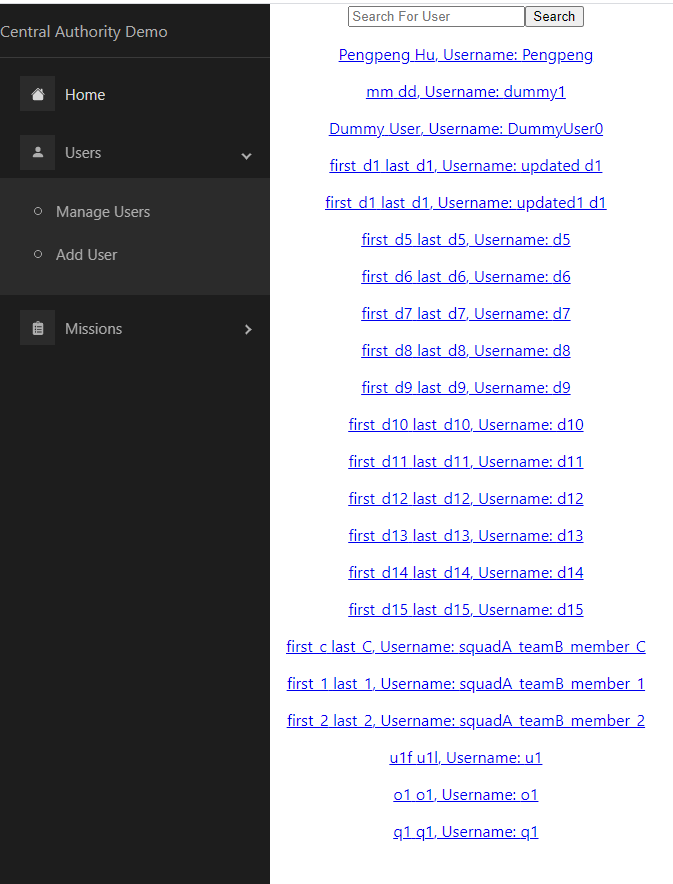


Figure 3. Shows all user information

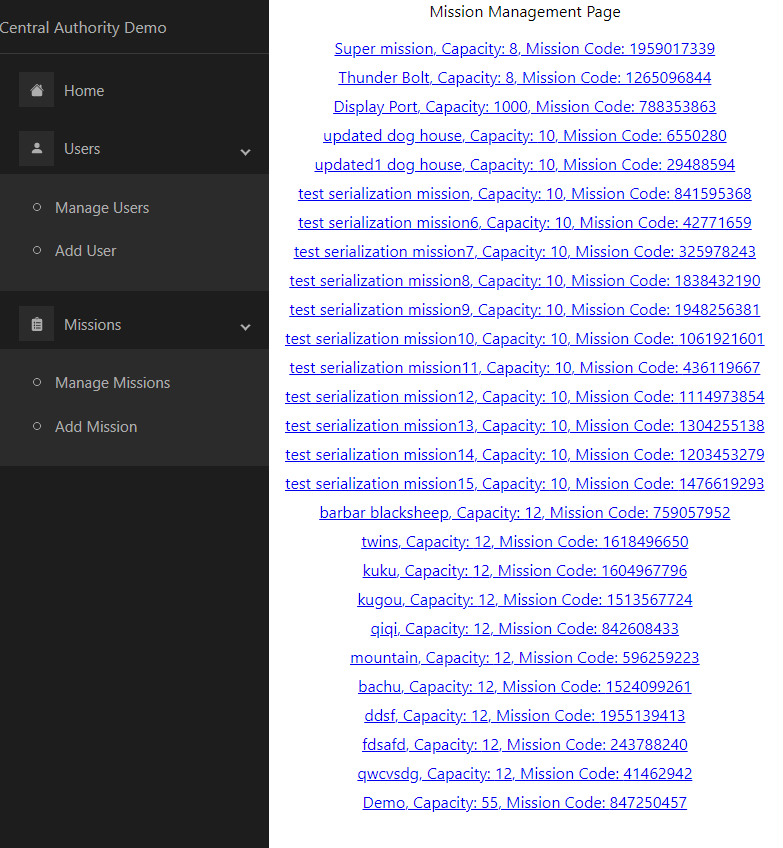


Figure 4. Shows all mission information

The manage mission page as shown in Figure 5 displays the mission detail including the PDF417 code of the mission code. We can modify this mission as well as adding new users to that mission. The newly added user will be assigned a tree ID which will be stored in the database. The user’s private key will also be generated and stored in the database. When the user request for a bootstrap, the central authority will fetch the keys from the database instead of generate in real-time which may cause delay.



Figure 5. Manage a specific mission, add user to that mission

**Restful API of the central authority:**

We have also implemented the backend as a restful API that takes HTTP request and response the required information as JSON file with HTTP response. The API is implemented with Java. The database of the API is MySql. The databased structure is as discussed in the report of June.

The API provide the interface to the following functionalities:

1. Get all the user information as JSON by request to (URL\_to\_server/Users and URL\_to\_server/GetUsersOfAMission)
2. Add user to the system by request to (URL\_to\_server/AddUser)
3. Add mission to the system by request to (URL\_to\_server/AddMission)
4. Add user to a specific mission by request to (URL\_to\_server/AddUserToMission)
5. Get user or mission count by request to (URL\_to\_server/GetUserCount and URL\_to\_server/GetMissionCount)
6. Get mission code as the pdf417 bar code by request to (URL\_to\_server/MissionQRCode)
7. Get all mission as JSON object array by request to (URL\_to\_server/Missions)
8. Search a specific User by request to (URL\_to\_server/SearchUser)
9. Update mission by request to (URL\_to\_server/UpdateMission)
10. Update user by request to (URL\_to\_server/UpdateUser)
11. Setup and transfer key for the user by request to (URL\_to\_server/Bootstrap)

The detailed request and output are shown in the figure below:

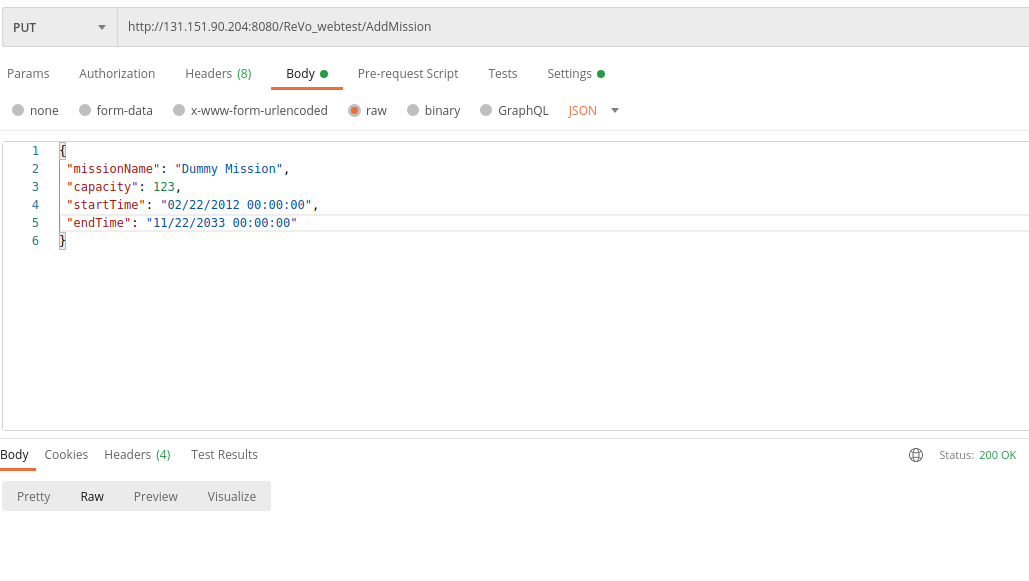


Figure 6. Add a mission to the system.

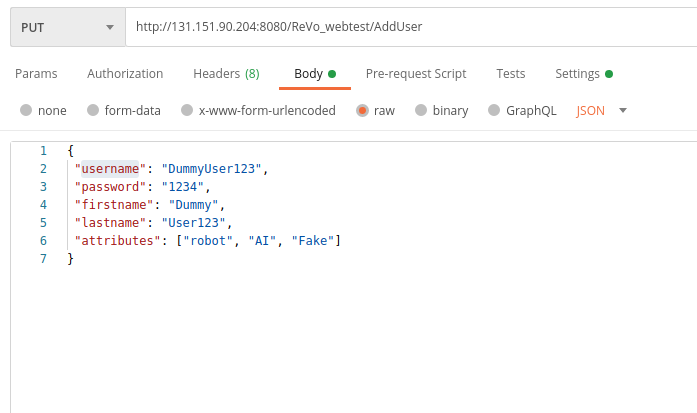


Figure 7. Add a user to the system

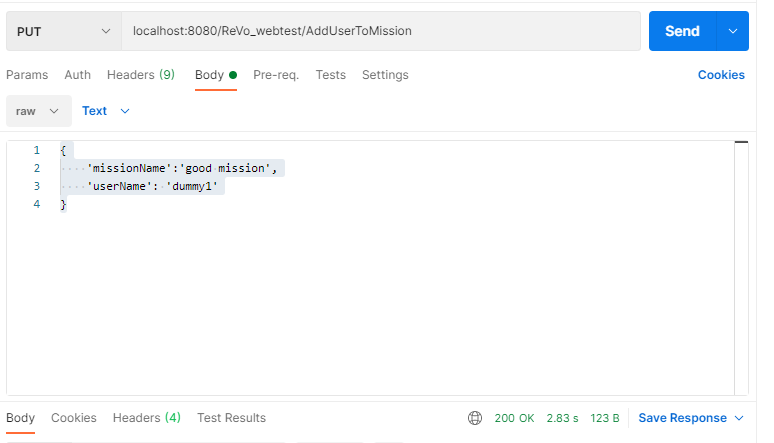


Figure 8. Add a user to a mission

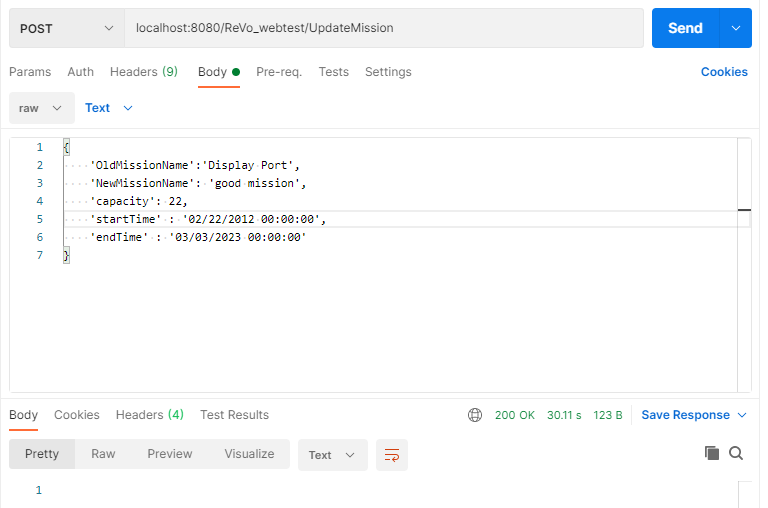


Figure 9. Update a mission

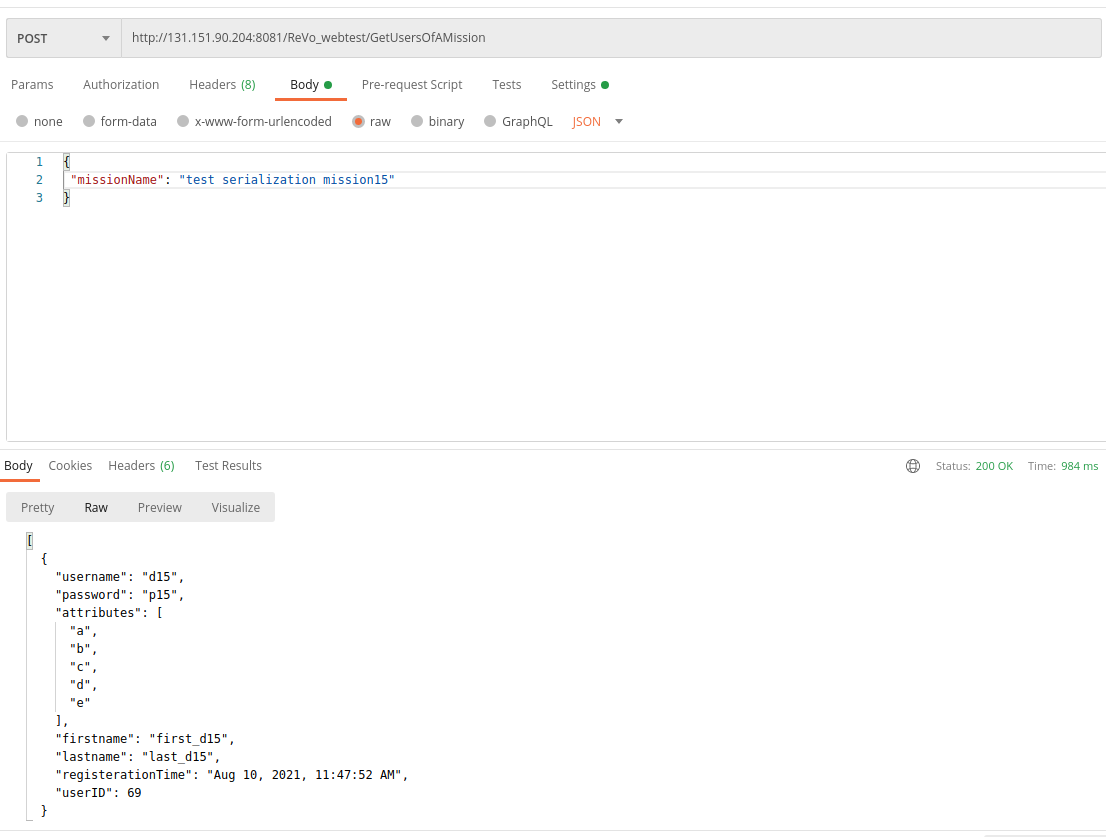


Figure 10. Get all users under a mission

The Figure from 6-13 shows the experimental result of request for different API. The request address are shown at the top box while the response of the restful API are shown in the bottom box. The right green text status shows the server response status. If the status shows 200 means everything is OK. We have implemented the status code based on the HTTP protocol.

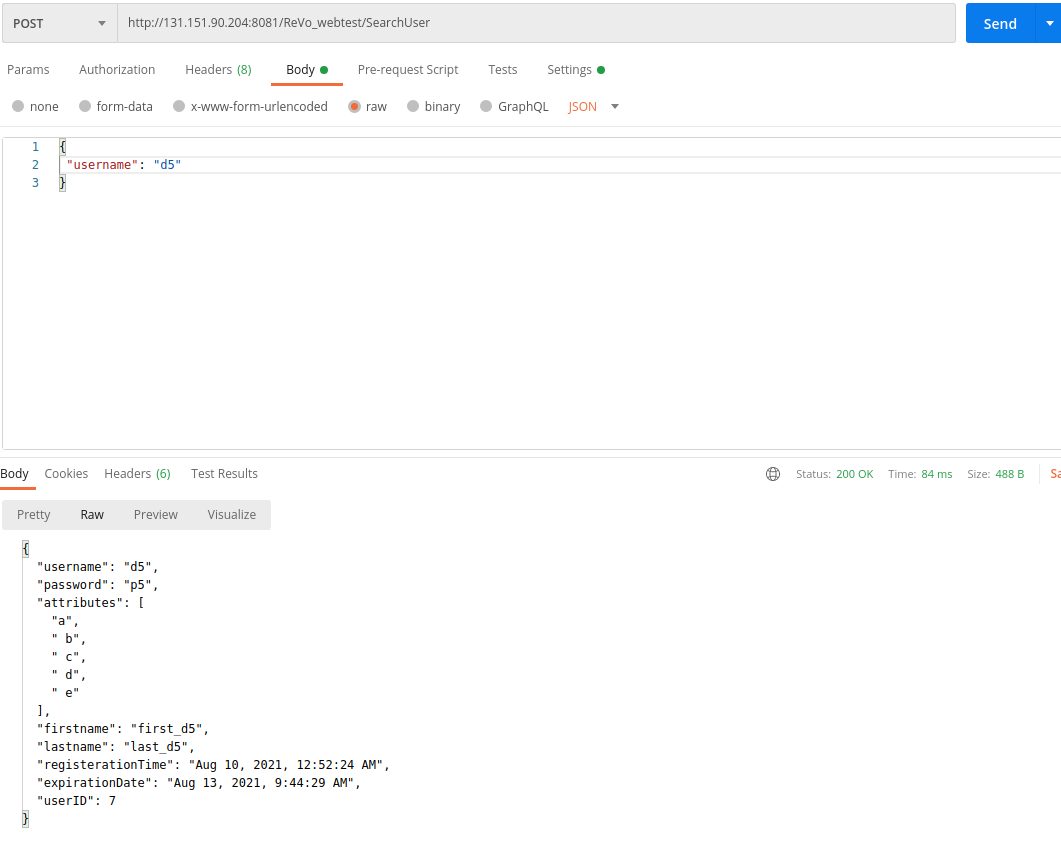


Figure 11. Find a specific user from the system

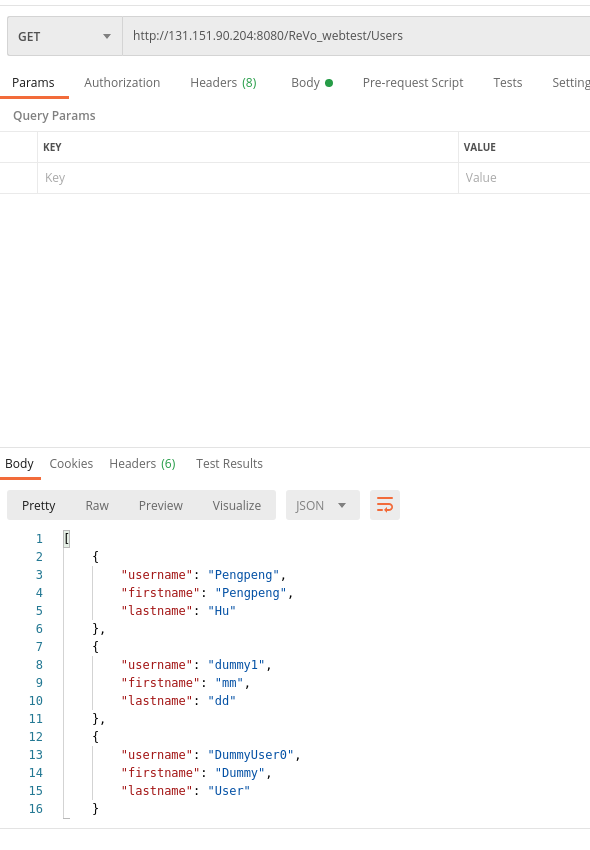


Figure 12. Get all the user information from the system

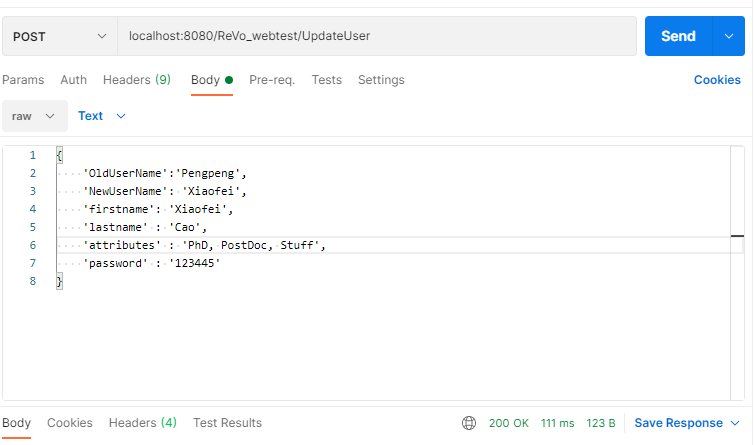


Figure 13. Update a user

**Conclusion:** We have implemented the front end and backend of the central authority. We have also integrated the Login/Logout page of the android application with the google API service page. We plan to integrate every next month.

1. **Content Caching for DTN network**

In this month, we have updated the Object 4 of our research. Here is the update.

**Objective 4:** Analyzing user trajectory for efficient caching: Users can be connected to different edge nodes at different times depending on their movement. Users request different types of content based on their interests. Hence, users' trajectory prediction enables edge nodes to efficiently cache content so that the users' interest will be satisfied, and the quality of service will be increased.

**Research Design:** Users, as well as the edge servers, are mobile, and hence different users are in the coverage of different edge servers at different times. In general, learning on caching is based on the prediction of future requests of content by the users. In our scenario, it also depends on the mobility of the users who requested the content. In a decentralized network, the delay of request and content forwarding can be significant due to the low connectivity of edge servers with the users and the command center. It is because edge servers collect and store data in the cache with the help of intermediate nodes. Similarly, edge servers forward the cached content to the targeted nodes via intermediate nodes. For efficient caching, it is necessary to analyze the historical mobility of the users and predicting their trajectory for the next short span of time. So that the edge servers can start caching the content according to the requests of the users who are most likely to meet shortly. For the training of user trajectories prediction, edge nodes always collect the metadata of the users' encounter information (i.e. encounter time, location). We can use the Long Short-Term Memory (LSTM) approach (which is an algorithm of Recurrent Neural Network) to train on user trajectory by dividing the network area as a grid which will utilize the time and frequency of a user being in a grid to find the next probable location of that user. Edge servers can also share their learning about the users with each other when they meet. Different edge servers operate in different zones, and due to the different mission goals, the nodes tend to roam around specific places for a certain amount of time. Hence, this situation can reduce the learning space of the edge servers about the nodes' trajectory.

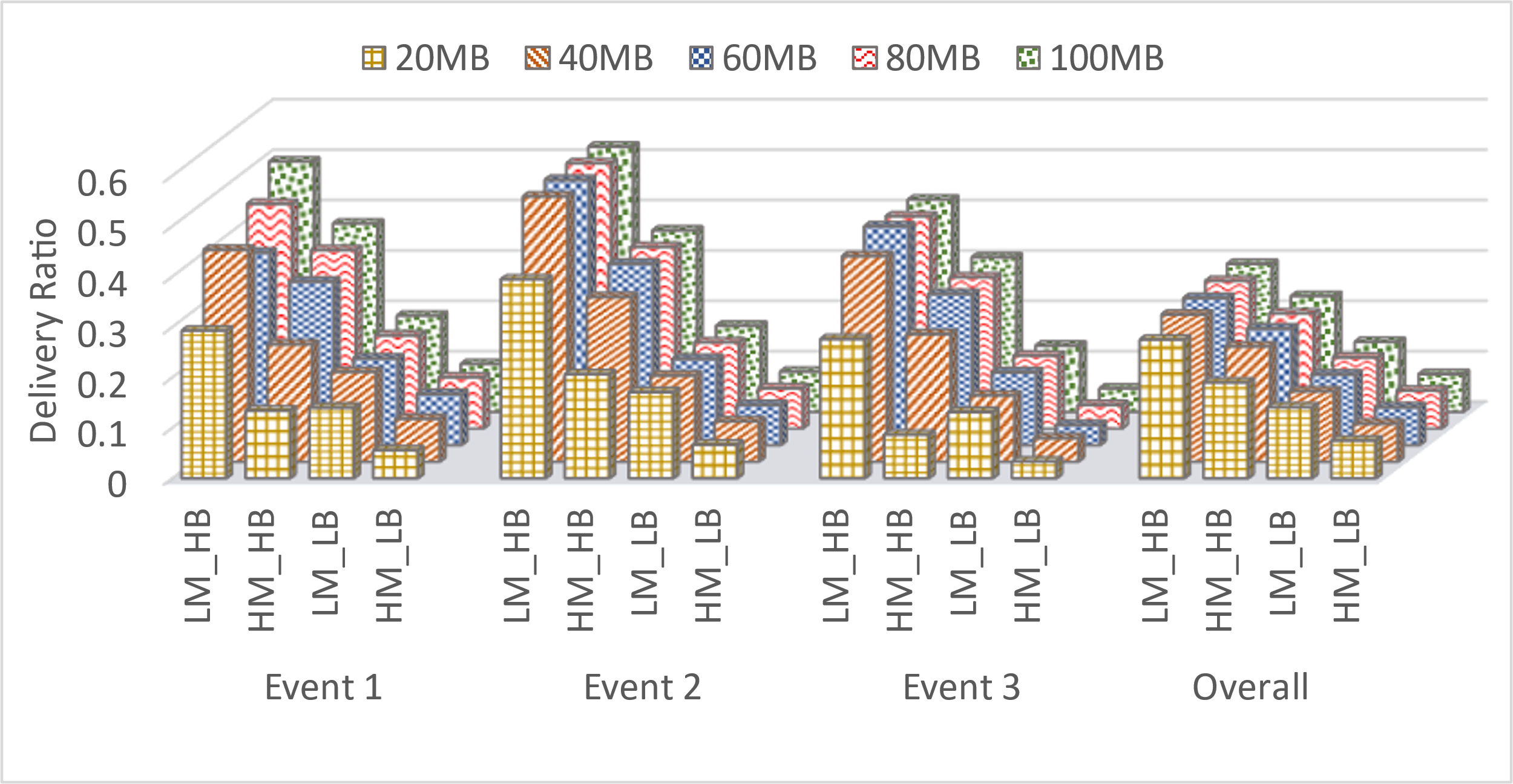
We prepare the trajectory data of the nodes by sending sample payloads to each other when any two nodes meet. This payload records the time when it is sent and received, the id of the sender and the receiver, and few properties of the sender node. Thus, we can record the time needed to forward data from one node to another nodes. This is explained with the example as follows: Assume node ‘a’ meets node ‘b’ and sends the payload. The payload will hold the record as <a, b, la, da t1, t2, hab> which means node ‘a’ sent a data to node ‘b’ at the time ‘t1’ and it was received at the time ‘t2’ after ‘hab’ hops, and the location and direction of node ‘a’ at time t1 were ‘la’ and ‘da’ respectively. Then, assume node ‘b’ meets node ‘c’ at time t3 and the payload from ‘b’ was received to node ‘c’ at time t4. It will create 2 records: <b, c, lb, db, t3, t4, hbc> and <a, c, la, da, t1, t4, hac>. Hence, the records include each possible data forwarding information for different pairs of nodes. This information grows exponentially. However, we control the growth by removing duplicate and old records. For example, a payload may have been received to a node from the same source node via different paths. In that case, we take the record which has the shortest delay and remove the other ones. Again, if the time difference of sending and receiving a payload is too long (exceeds the ttl of a message) we remove the record. Besides, if the hop count of a record is too long we remove the record. Finally, those records are analyzed to predict the delay of sending data from one node to another given the location and direction of the source node at a certain time.

**Simulation Setup:** We perform simulations on ONE simulator with 131 nodes. These nodes are mobile and capable of transferring messages when they are within the communication range. We use the following two datasets for the simulation.

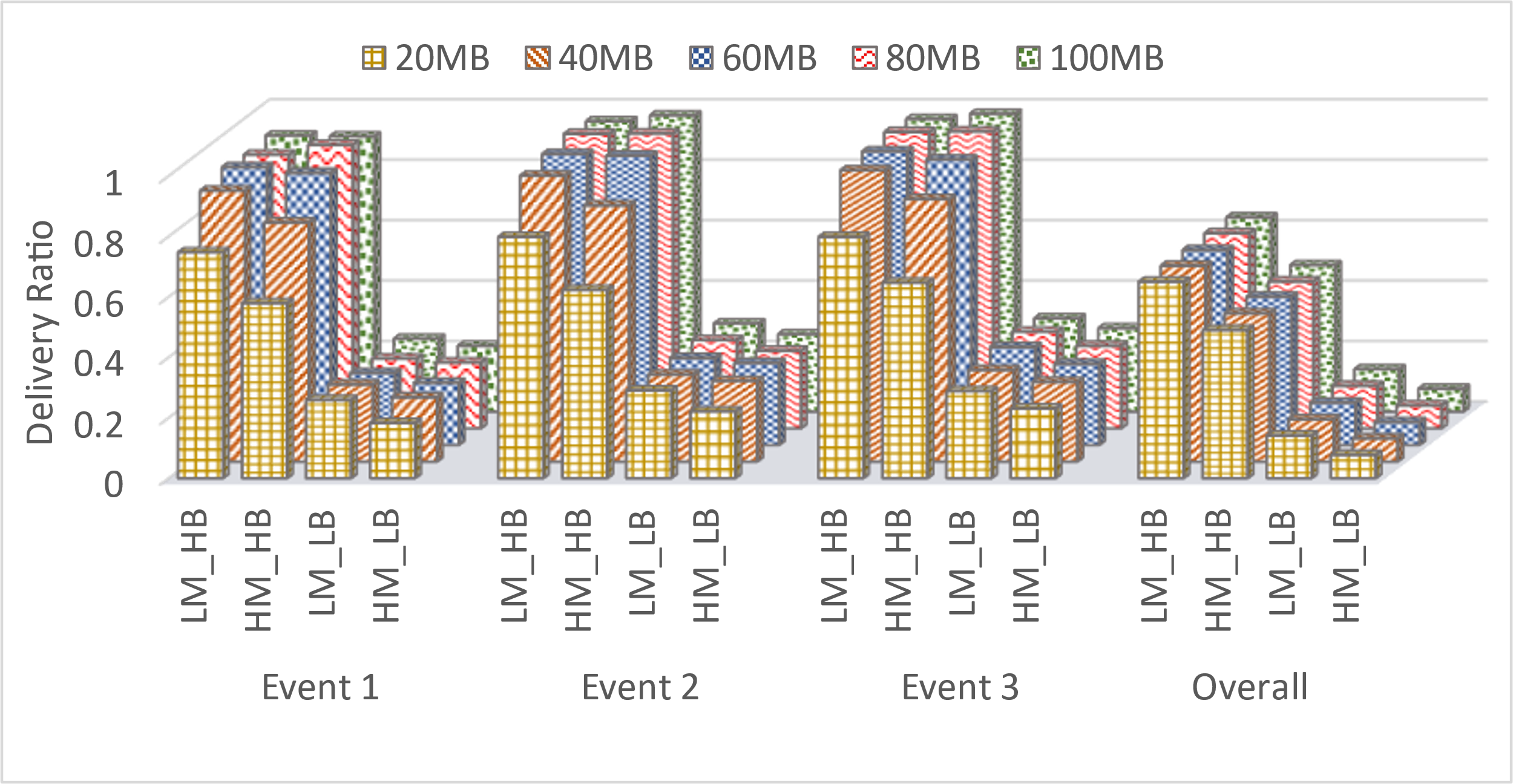
* Asturies dataset (AST): This dataset contains one year long GPS traces collected from the regional Fire Department of Asturias, Spain. The data set depicts a disaster environment like sparse communication and heterogeneity, and thus, it is very important to manage content efficiently in this environment. Due to the lower rate of connections among the nodes in this dataset, we divided the total time of the dataset by ten, and then took three days of data for the simulation which is a month equivalent of traces. Again, due to the sparsity of the nodes, we scale down the traces by a hundred and consider a (1000 x 2100) m2 area where nodes are relatively congested from the remaining part of the network. To simulate events in this dataset, we choose 3 different points (550,550), (250,1550), and (550,1350) as the locations of 3 events that occur in the 18th, 36th, and hour of the event generation time, nodes within a 100-meter radius of the event location create many messages with the subset of topics from the event topic set.
* EPFL mobility dataset (EPFL): This dataset contains mobility of 500 taxis over 30 days in an Francisco Bay Area, USA. Due to the slow progress of nodes, we divided the total time by 10 and use two days of data in our simulation which is 20 days equivalent of mobility. Then we filtered the most populated area of the dataset and scaled-down the traces by twenty to increase the contact and finally considered a (4000 x 4000) m2 area. We filtered nodes with less mobility and kept only 131 nodes to match with the AST dataset in our simulation. The locations of the three events are set as (1900,1500), (1900,1300), and (2900,1300) points where nodes are more populated in the 12th, 24th, and 36th hours of the simulation respectively. Nodes within a 200-meter radius create event messages during the events.

In the following figure, we show the changes in the delivery ratio with the changes in network resources. With the increase in network bandwidth and nodes buffer size, more messages are delivered. Again, if the number of messages increases, more messages are delivered but the ratio of the number of messages delivered vs. the number of messages created, i.e. the delivery ratio decreases. For a lower number of messages, the increase in delivery ratio from the larger to higher buffer size is more when the bandwidth is high (20%, 13%, 14%, 2% for events 1, 2, 3, and overall messages respectively for AST dataset, and 17%, 16%, 17%, 0.7% for EPFL dataset). Similarly, for a higher number of messages, the delivery ratio increases from the lower to higher buffer size significantly when the bandwidth is high (23%, 15%, 22%, 4% for events 1, 2, 3, and overall messages respectively for AST dataset, and 33%, 36%, 35%, 0.5% for EPFL dataset).

Overall, bandwidth plays a significant role in increasing the delivery ratio for EPFL dataset as the total number of created messages is less than the AST dataset. As a result, more percentage of overall messages can be disseminated for the EPFL dataset when bandwidth is higher.



a) AST



b) EPFL

Figure 14: *Effect of varying network bandwidth, number of created message, and nodes’ buffer on delivery ratio. Here, HB (Higher Bandwidth) = 2MB/s, LB (Lower Bandwidth) = 128KB/s, and HM (Higher Message) = 36997 & 27078, LM (Lower Message) = 17889 & 13758 for AST & EPFL dataset respectively.*

We will use these two datasets for generating graphs for the above-mentioned 4 objectives.

* Delivery ratio of event messages with different trending threshold:

In the following graphs, we vary trending threshold Ɵ and provide the result of the delivery ratio of event messages. We see that, the delivery ratio is higher in EPFL dataset as nodes are less sparse. Besides, delivery ratio increases when we remove the constraints of resource (Epidemic routing). However, the highest average delivery ratio (28.4%) of all the events combined for the AST dataset is achieved when Ɵ = 15. Therefore, we maintain this value in all of our simulations for this dataset. For the EPSL dataset, the highest average delivery ratio (68.7%) of all the events combined is achieved when Ɵ = 60.

1. AST
2. EPFL

Figure 15

* Congestion effect during events:

In the following graph, we show the effect of event messages delivery ratio when learning is incorporated considering congestion. In our method (see figure 5a), we see that with the congestion learning (CG T), more event messages are delivered than CG F scenario (2%, 0%, & 1% more on average for events 1, 2, and 3, respectively for AST dataset and 2.3%, 1%,

& 6% more for EPFL dataset) which means the learning can help in delivering more messages in a congested environment.

1. AST
2. EPFL

Figure 16

1. **Dependency Tree and Parse Tree Generation**

This past month I worked on finding equivalent java libraries to replace python libraries, as we are integrating our code to Android and android runs on java. Also, I increased the size of the dataset from 200 images to 300 images by adding images of burning cars, and only soldiers pics. I will discuss more details about it in the following sections.

**1. Finding equivalent java libraries to replace python**

The below image which is a screenshot of a code snippet that generates captions for an image displays the libraries that are used to generate a caption for an image.

I have been focussing on finding equivalent libraries in java, firstly for TensorFlow, [TensorFlow Java](https://github.com/tensorflow/java) can run on any JVM for building, training and deploying machine learning models. It supports both CPU and GPU execution, in graph or eager mode, and presents a rich API for using TensorFlow in a JVM environment.



Figure 17: libraries used in generating captions

To illustrate the importance of TensorFlow, let us take a look at the various parts of code where TensorFlow is used.

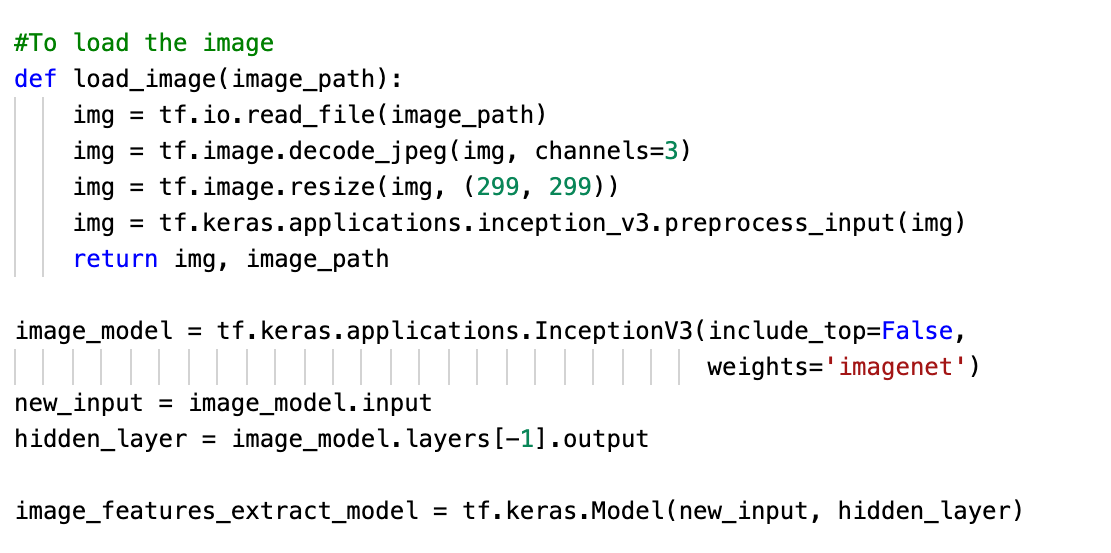
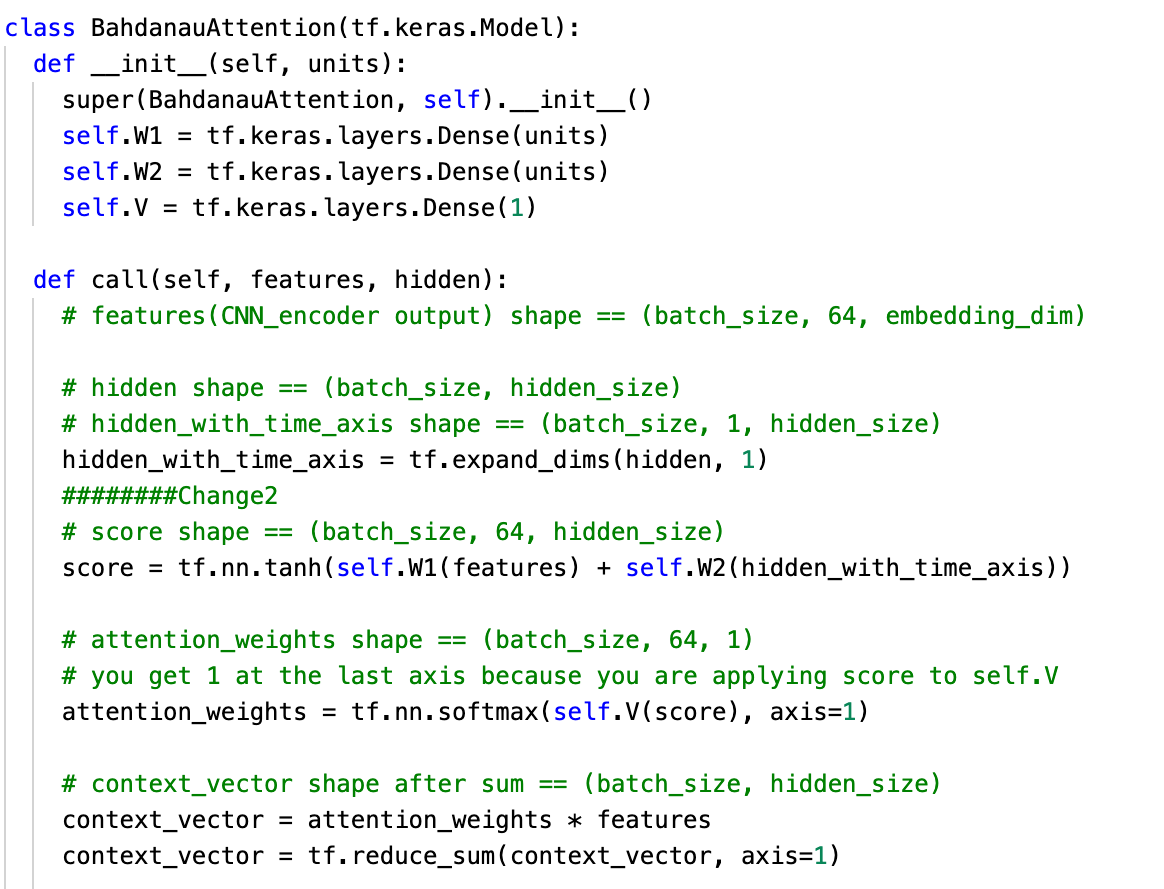


Figure 18: TensorFlow usage while loading an image

Starting from a basic task to load an image and to use its functions like resizing it, decoding it. It’s easy in Python but in java, it’s a different story altogether.

TensorFlow is also used in BahdanauAttention class, where we create the layers of the model, specify the softmax functions, expand dimensions. All this becomes a herculean task while redoing it in java.

Figure 19: TensorFlow usage in BahdanauAttention

Even in the CNN encoder class, A Convolutional (CNN/CNN)-based Encoder-Decoder Neural Network is an encoder-decoder neural network that consists of an encoder neural network and a decoder neural network in which one or both are convolutional neural networks.

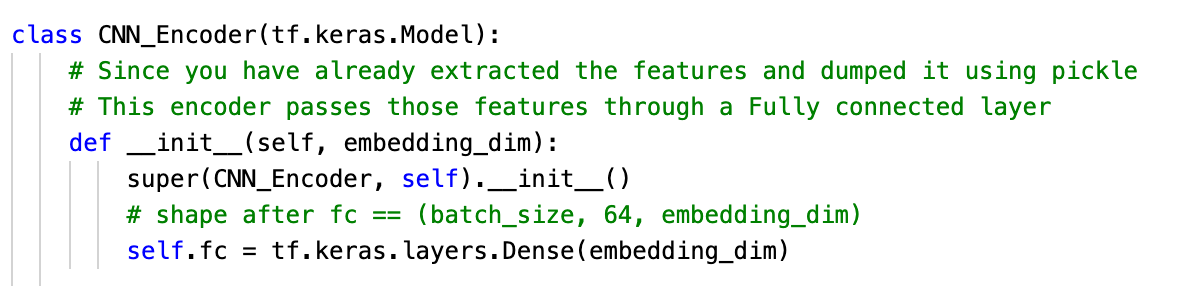


Figure 20: TensorFlow usage in CNN\_Encode

Finally, to evaluate an image like below we use libraries such as NumPy and TensorFlow.



Figure 21: TensorFlow usage in evaluating an image

Another important library to which I am finding a replacement is for the NumPy library, NumPy is the fundamental package for scientific computing in Python. NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences.

A somewhat similar library I found is JavaNpy, which enables loading of n-dimensional arrays in java, therefore Android. Here’s the link to that library <https://github.com/dreamolight/JavaNpy>

Another huge library that needs replacement is sklearn, Scikit-learn is probably the most useful library for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction. For this, I am still trying to find a proper equivalent.

**2. Increasing Dataset size**

Increased the dataset size with the following images and captions related. Included the images of soldiers, and burning cars.



Figure 22: Burning car on the battlefield



Figure 23: Burning car on the battlefield



Figure 24: Soldiers



Figure 25: Soldiers

**3. Future Work**

Convert the python code to java, eventually to Android, that goes through line by line of how the libraries are being used and replace it with equivalent java code. Find a similarity score based on interesting words.