**Quarterly Status Report**

**Report for period: Sep 01st, 2021 – Sep 30th, 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**

**Secure data sharing**: The implemented attributes based encryption algorithms requires two kinds of security validation for success decryption. First, the eligible user should contains valid attributes that satisfy the encoding policy. Second, the eligible user shouldn’t be in the revocation list. When both of the validation requirements are met, the user can decrypt the encrypted photos and texts. Otherwise, there are the following three invalid situations:

First, a user is a member of a mission and he received an encrypted message from the member of the same mission. The user is not in the revocation list. However, the user’s attributes not met the minimal validation requirements of the policy. Then the user won’t be able to decrypt the message. In the APP, the encrypted photo will not be shown. Instead, a warning thumbnail will be displayed indicate the user received a message which is not for him as shown in Figure 1. However, the APP will keep the encrypted photo and try to pass it to the eligible mission members.

Second, a user is a member of a mission and he received an encrypted message from the member of the same mission. Also, the user’s attributes met the minimal validation requirements of the policy. However, the user is in the revocation list. Then the user won’t be able to decrypt the message. In the APP, the encrypted photo will not be shown. Instead, a warning thumbnail will be displayed indicate the user is revoked the right to receive this message. A warning message will be displayed on the screen as shown in Figure 2. The APP will then discard the encrypted message and not store any information of that message in the current revoked device.

Third, a user is a member of a mission but he received an encrypted message from a member from another mission. Although his attributes may meet the minimal requirements of the mission and he may not be in the revocation list, he will still not be able to decrypt the encrypted message and photo. Instead, a warning thumbnail will be displayed indicate the user received a message which is not for him as shown in Figure 3. However, the APP will keep the encrypted photo and try to pass it to the eligible mission members.



Figure 1. Warning thumbnail of invalid attribute.



Figure 2. Warning thumbnail of revoked user



Figure 3. Warning thumbnail of invalid permission

**Mission members page:** Last month, we also implemented the mission members page which displays all the members nickname in the same mission. Each row of the list contains the user’s nickname and a switch. By toggling the switch, the user could easily revoke or not revoke a member. The page use a shared view model to store the revocation information and share the information with the security service. Figure 4 shows an example of this page.

When a user is be toggled, the user is revoked temporarily from the sender’s revocation list. When the receiver received an image which indicates he is been revoked, the user won’t be able to decrypt the image. The revoked user won’t be able to hold any image also. However, when the sender toggle that user again, the user can back from the revocation list and the user’s status becomes OK. Then the user can receive and decrypt the image again.

**Revise and optimizations:** Last month, we also made the following revise and optimizations. First, we have revised the UI of the image sharing. We remove the text field for manually enter the revoked node ID. Instead, we have created a dedicated page to show and revoke members in the mission. Second, we have optimized the data transmission flow. Now, user can view the warning thumbnails if they are not able to decrypt the receiving photos. Third, we have optimized the font which makes the text more readable.

**Future work:** Next month, we will create a working demo shows the secure data sharing.

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.



Figure 4. Dedicated revocation list

**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.

To fit the trajectory data for learning with LSTM, we create time-series data from each node’s trajectory by sampling the location of a node every fifty seconds. If data for a certain time range is not available, then we create additional rows with the missing time windows using the last available location from the dataset. Data is then copied to multiple consecutive samples where each sample includes 180 training data and 90 testing data. There are overlaps on the sample sets so that necessary information (i.e. trajectory sequence) is not missed. The job of the LSTM algorithm is to predict a sequence of grids at a certain time range. In our scenario, it predicts 75 minutes of location grids of a node from the previous 150 minutes of trajectory data. However, predicting nodes’ trajectory doesn’t help in predicting if a node can forward a message to some other node. It is because two nodes may not be in the same grid at a certain time although be able to forward messages via different intermediated nodes. This problem can be expressed as a graph algorithm where nodes are the vertices of the graph and edges between the nodes are the time window they met at some location. From the graph, a node can check if a path exists to forward a message to some particular node. Besides, from the edge, the time needed for the message to arrive can also be measured. This algorithm depends on the combination of the accuracy of the nodes’ predicted trajectories. Moreover, in DTN, nodes may not have the availability of the other nodes' prediction model. In addition, the availability of other nodes’ models may not always help as the current sequence of the location of other nodes may not be updated due to the lack of contacts. However, on a battlefield, particular places such as camps, shelters, military bases, water sources, etc. function as the meeting places of different nodes operating on different missions. Nodes roam around in those places for more amount of time. Therefore, we find out the popular grids in our scenario where more nodes visit during the day. Then we update the dataset as: a node after entering one of these important grids, we make it stop for a certain amount of time. This increases the message forwarding possibility and also relates to the real scenario.

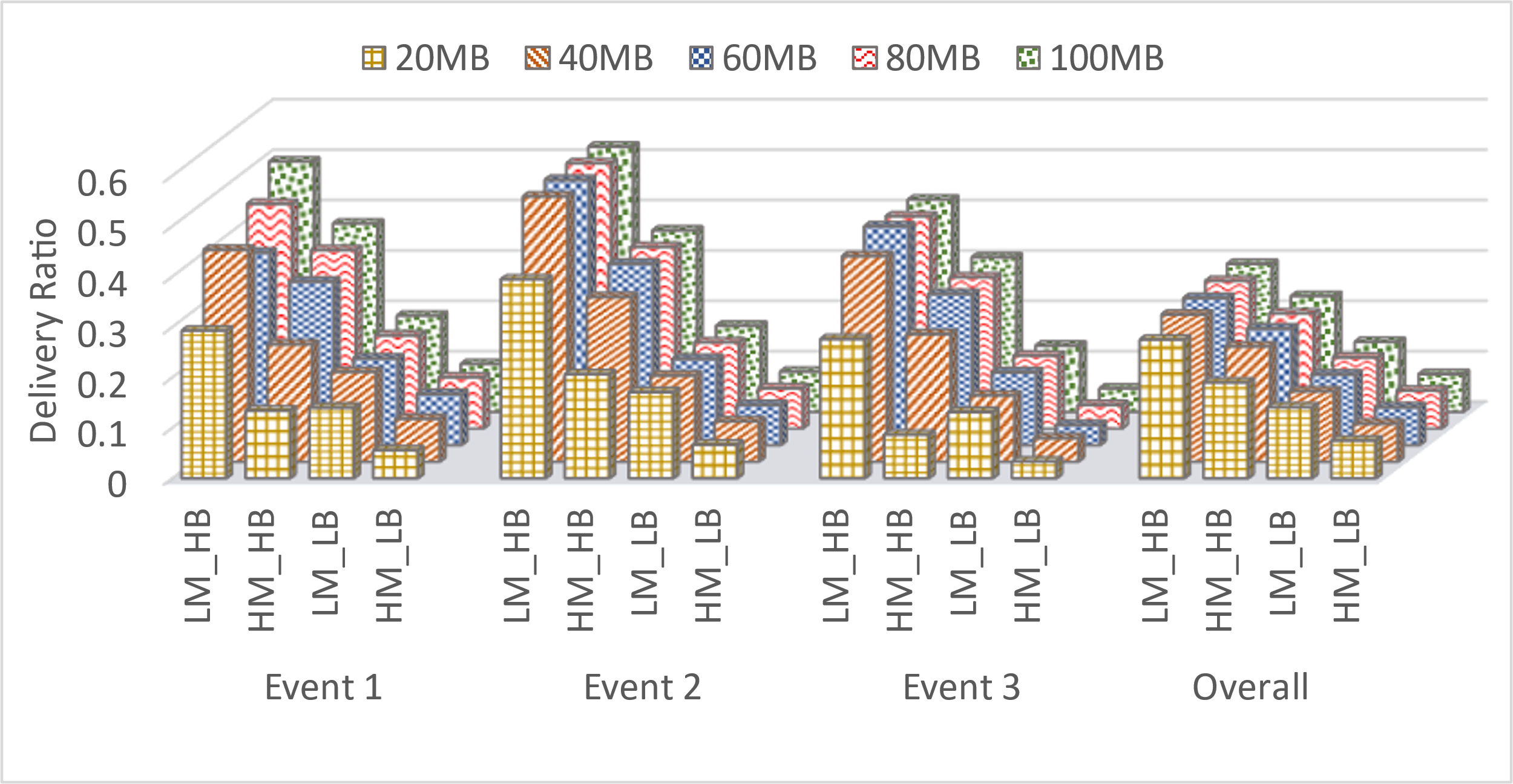
To verify the accuracy of our prediction, we prepare the validation data by sending sample payloads to each other nodes when any two nodes meet. This payload records the time when it is sent and received, the id of the sender and the receiver, and few properties of the sender node. Thus, we can record 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 validate that if sending data from one particular node to another is possible within 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.

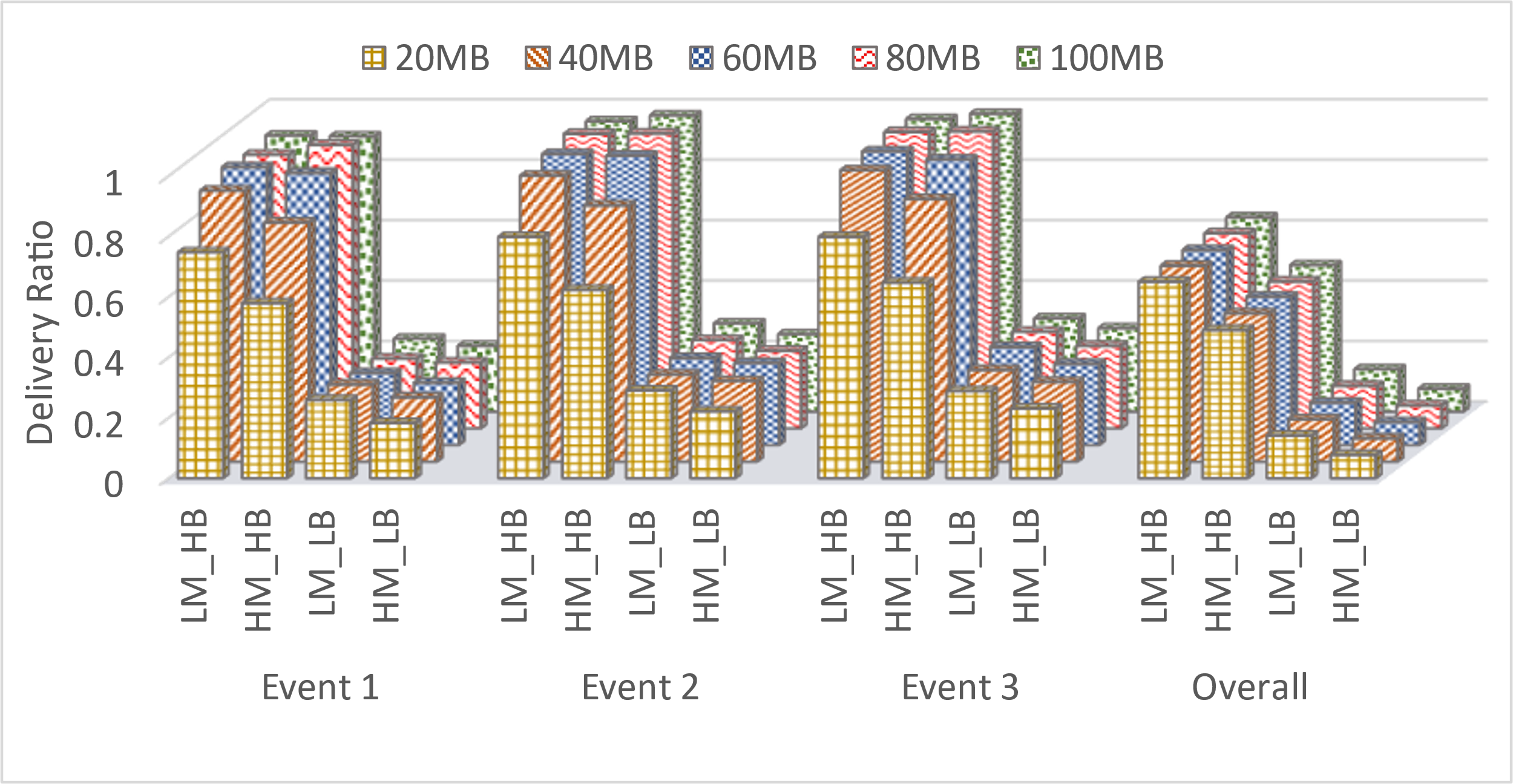
* Asturies dataset (AST): This dataset contains one year long GPS traces collected from the regional Fire Department of Asturias, Spain. The data set depicts a disaster environment like sparse communication and heterogeneity, and thus, it is very important to manage content efficiently in this environment. 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.

θ =

1. AST

θ =

1. EPFL

Figure 5

* 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 6

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

This past month I worked on finding Image Analysis libraries and I was able to explore Microsoft’s cognitive services API’s in detail. Several other image analysis libraries I found useful are google’s cloud vision API, cloud sight, Clarifai and Amazon Rekognition. Let’s discuss in detail Microsoft’s cognitive services API and the limitations of using it with Delay Tolerant Networks.

**1. Azure’s computer vision service**

Azure’s computer vision service gives access to advanced algorithms that process images and return information based on the visual features we are interested in.

There are various services that azure offers namely Image Analysis, Optical Character Recognition, and Spatial Analysis. The service which is of interest to us is Image Analysis.

It can basically accept images in any format, file size must be less than 4 MB though. Sample response of image analysis is provided below:



Figure 7. Sample image

**JSON response:**

{

"description": {

"tags": ["outdoor", "building", "photo", "city", "white", "black", "large", "sitting", "old", "water", "skyscraper", "many", "boat", "river", "group", "street", "people", "field", "tall", "bird", "standing"],

"captions": [

{

"text": "a black and white photo of a city",

"confidence": 0.95301952483304808

},

{

"text": "a black and white photo of a large city",

"confidence": 0.94085190563213816

},

{

"text": "a large white building in a city",

"confidence": 0.93108362931954824

}

]

},

"requestId": "b20bfc83-fb25-4b8d-a3f8-b2a1f084b159",

"metadata": {

"height": 300,

"width": 239,

"format": "Jpeg"

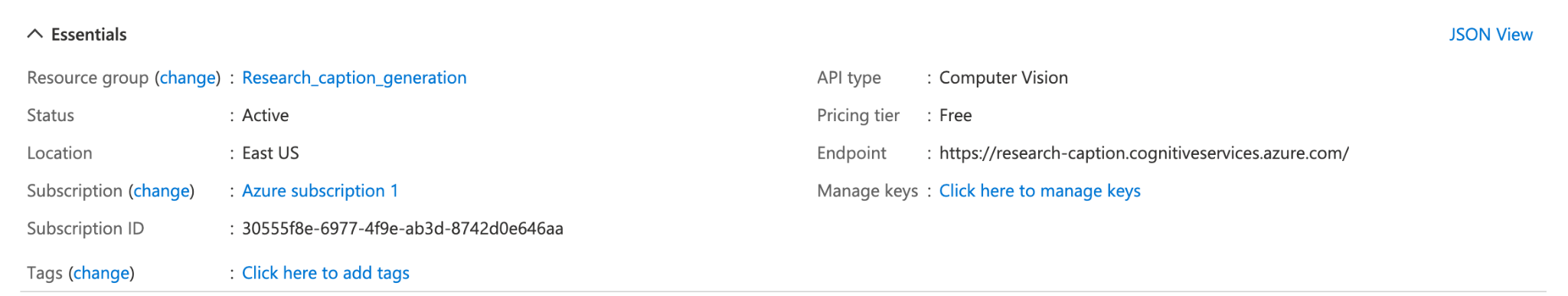
}

}

As shown in the former part, for a sample image with objects such as buildings, sky and so on, the Microsoft cognitive service API gives a response in JSON format with the details as shown above.

The cognitive service API is a large repository that consists of several endpoints, among which Image Analysis API endpoints are what we consume to get descriptions, tags, objects and several details of an image. Let’s have a briefer look at the code snippets and also details of a quick start to using the image analysis client library or REST API.

Firstly, we need to have an Azure subscription that offers a free one month trial, once we have an Azure subscription. We need to create a Computer Vision resource in the Azure portal to get the key and endpoint.



The endpoint we are supposed to hit is:

<https://research-caption.cognitiveservices.azure.com/>

Then I went ahead and included the maven artefact in the pom.xml to use this client library as shown below.

Once we have the azure computer vision client library, the key and endpoint. We can call the service from a local java code and get an analysis of an image. Figure 8 is the screenshot of the sample java code.

As shown in Figure 9, I have passed a local image path to a post request made to an endpoint. The below image shows, the features and specifics of an image we can get using Azure cognitive services.

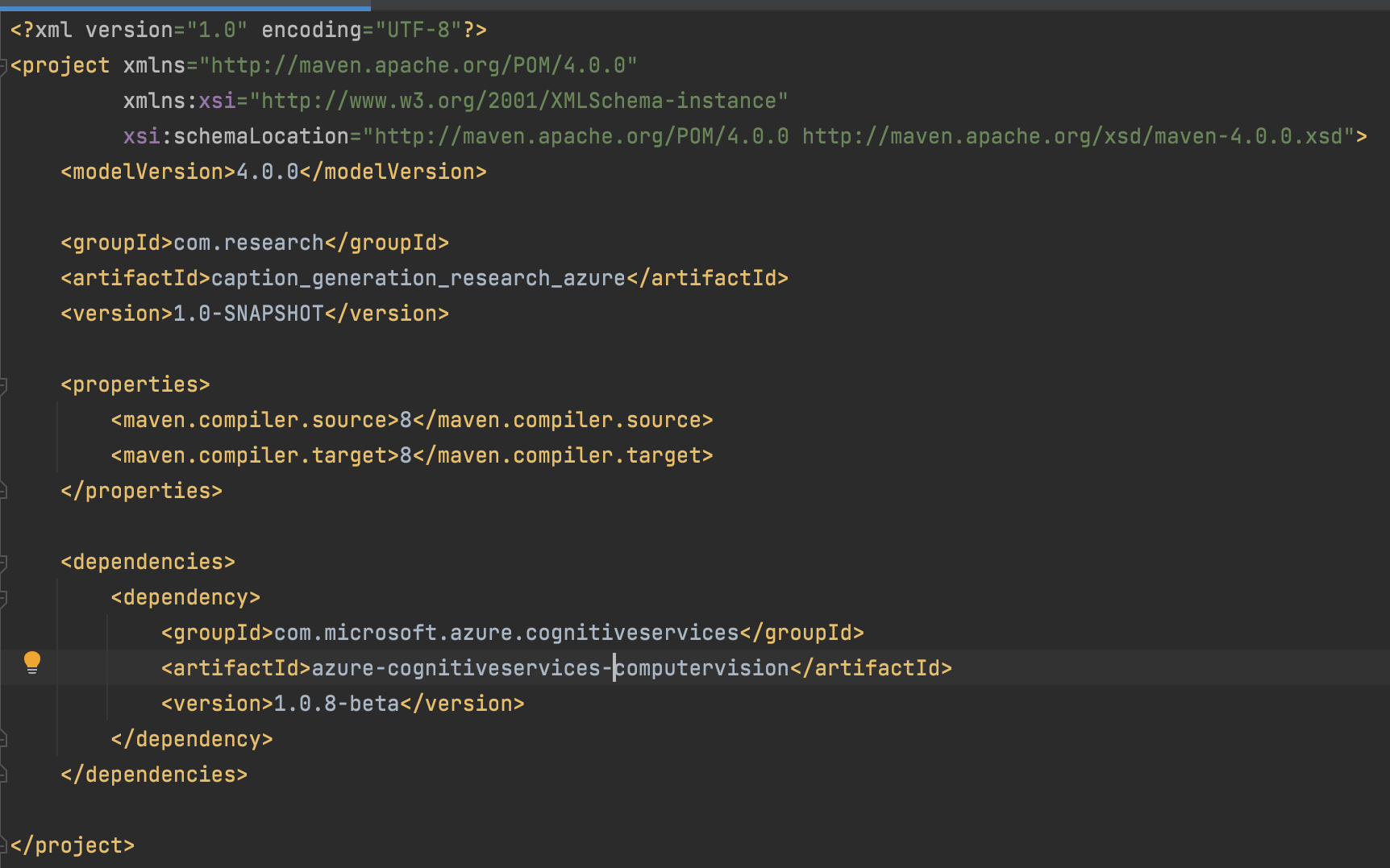


Figure 8. pom.xml



Figure 9. Sample Java code

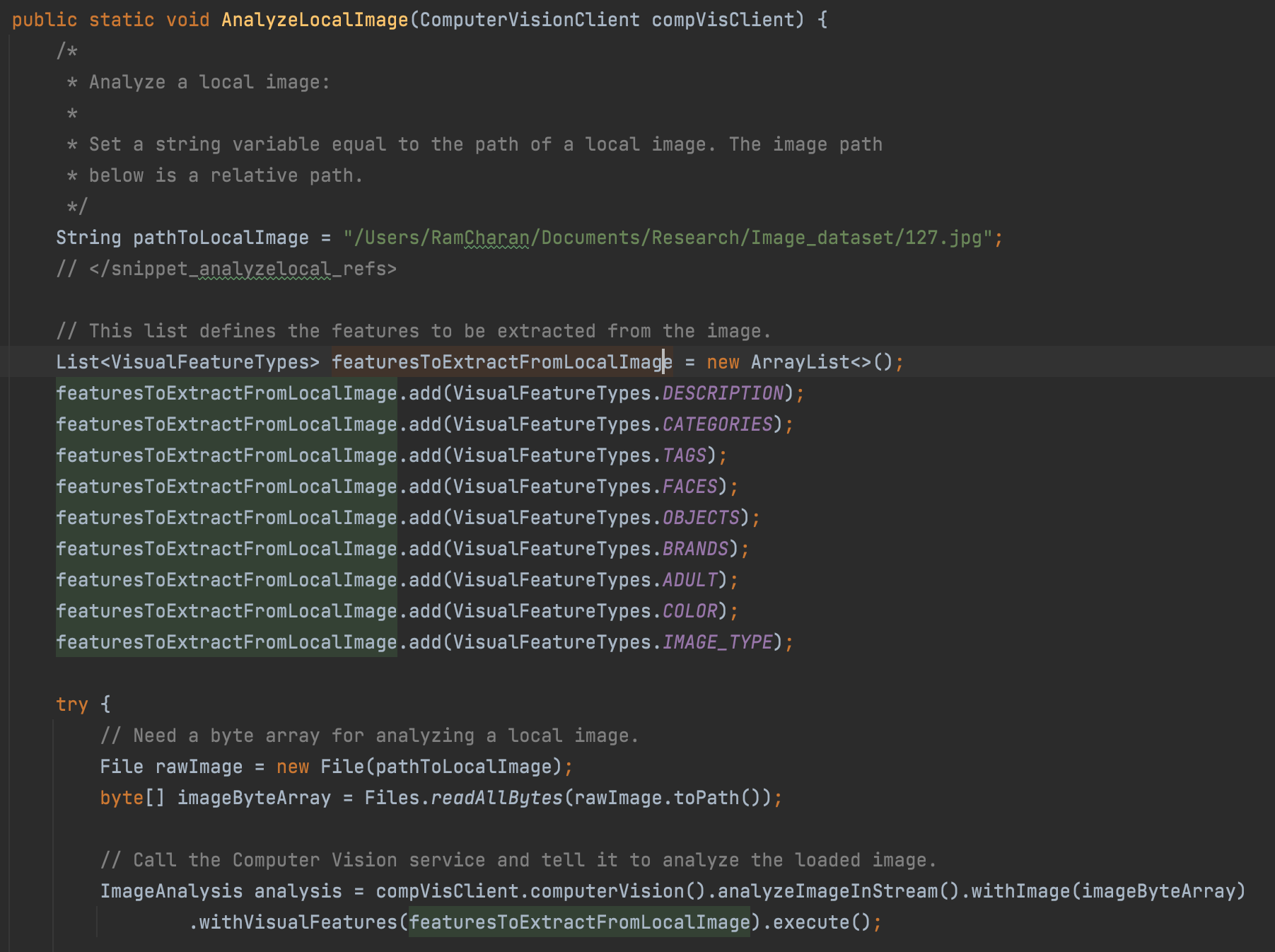


Figure 10. Sample Java code 2

We can expect visual features such as description, categories, tags, faces, objects, brands, adult, colour, and image type.

**2. Limitations of using it with DTN:**

Although it provides specifics of an image in detail, we cannot use it on the battlefield because we are using Delay Tolerant Networks. Delay Tolerant Networks works on low energy and bandwidth, which means they do not use the internet to transfer or request any kind of data. Hence this restricts us from using any of the cognitive computer vision API services.

Even though we cannot use it, it is a great find because we can use it in our test setup for generating tags, descriptions automatically.

**3. Future Work**

Since we cannot use the cognitive computer vision API services, we are supposed to train the model and use it to generate photo descriptions. Hence, I will go back to the existing model I trained with my custom dataset and use it for further automatic photo descriptions. Also, compare two images based on their parse trees to find the similarity.