International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 9 (2018) pp. 7168-7177

An Effective Simulation Model for Optimal Traffic Flow across Packet Data Network

Nagaraju Baydeti[a*], Mariappan Vaithilingam[b], Ramachandran Veilumuthu[c]

[a] *Department of Computer Science and Engineering, National Institute of Technology Nagaland, Dimapur 797 103, India.*
[b] *Software Development Manager, Amazon Bengaluru, India.*
[c] *Department of Information Science and Technology, College of Engineering Guindy, Anna University, Chennai, India.*

** Corresponding author.*
(Nagaraju Baydeti)
(Ramachandran Veilumuthu)

Abstract

The primary aim of this research is to reduce the routing of redundant data across social media by means of a novel procedure which capitalizes the availability of the requested content, based on extraction and comparison of frames especially for the video content sharing. This model checks and validates the extracted frames with the existing contents during the initiation of any data transfer and takes a decision whether to handle the push or pull operation. Exclusive study of this approach is carried out by simulating a real-time content sharing environment using Python as front end and MySQL as backend with fifty end users each one trying to share video contents randomly between 200 to 2500 initiations at arbitrary time intervals. The simulation results show that by exploiting the existence of contents prior to the initiation of transferring or receiving brings out significant reduction of redundant data traffic across various planes of mobile communication environment via social media leading to the optimization of the operational costs in a network and effective utilization of data resources.

Keywords: Social Media; Video Sharing; Data Traffic; Simulation.

INTRODUCTION

In the recent past, mobile and wireless data communication have evolved tremendously, one of the reasons for this rapid growth is due to the substantial data exchange across smart devices. The data exchange between the personal computers during late 90's is now dominated by the smart devices around the globe. Additionally, the concept of wireless sensors, smart grids, smart devices, smart home and smart cities with respect to the dawn of Internet of Things (IoT) where every device resulting in the transmission of data is acting as a catalyst for the increase in huge volume of data exchange which includes video streaming, web based data transmission and messenger services through social media. In this scenario of huge evolution in the data traffic due to revolutionary increase in the growth of IP based connected devices, it is expected that beyond 2020, handling huge volume of data traffic is a key challenge for next generation mobile networks. Further, starting from the introduction of mobile data services i.e., from 2.5G–GPRS, 2.75G–EDGE, 3G, LTE, LTE-A to the upcoming 5G, the provision of higher data rates is one of the Key Performance Indicators (KPI) to evaluate the standards. Soon, it becomes inevitable to have a seamless transfer of data with minimal latency where every device needs to be well connected and exchange of data instantaneously with highly reliable Quality of Service (QoS).

The introduction of relatively low cost smart phones having multi-core processing units with gigabytes of memory, high resolution digital cameras and affordable mobile applications enable the user to create their own multimedia content for sharing in the social media. Additionally, extensive data rates accomplished by modern generation mobile communication standards like LTE, LTE–A lead to the densely populated video streaming applications. Moreover, due to the popularity of the content, trending videos are shared extensively among the users who are geographically distributed across the globe and connected by the federation of Mobile Network Operators (MNOs).

In the existing models [6, 7, 9], it is inferred that for every request / response, the relay of data packets takes place in the Internet layer for identical and non-identical content transmission, which is a bottleneck effecting the degradation of performance metrics in the network. Further, in the instance of identical content request from the devices of heterogeneous MNOs, the response will be relayed in multiple routes based on the MNOs infrastructure, which might lead to congestion and redundant data traffic flow at the packet data network layer. However, the Internet's first-generation protocols, which were originally developed to provide connectivity to various computers from different networks for routing the IP based traffic are now facing a tough challenge to cope up with the pace of users' requests in the contemporary digital era and to sustain the massive data load which has to be delivered with reliable and high data rates. Considering the immense raise in the number of connected IP based devices and the identical traffic flow, a novel approach is proposed for regulating the capabilities of the Internet layer to optimize the flow of data traffic.

At present, most of the data traffic originated from the users of smart phones and other smart devices are due to the sharing of popular contents. The tradition of sharing popular contents across social media has experienced profound growth from the last decade. Video sharing is one of the most popular scenarios

in the field of communication in which the trending videos are shared among several users simultaneously over a period, which leads to increase in the data traffic across packet data network. Heaps of frequent identical content sharing among the users lead to multiple copies of similar packets routed across Internet plane resulting to traffic congestion and if not handled properly becomes a bottleneck for the service providers with respect to the data rates.

The rest of the paper has been organized in sections of which, the related work in section 2 covers the aspects of caching the contents in the various planes of mobile communication network to avoid the delay in transmitting the contents to the end user. Section 3 details the proposed model and algorithm for optimal traffic flow in social networks. The experimental real-time data traffic simulation is detailed in Section 4 and the simulation results are described and discussed. Section 5 concludes with the outcome of the proposed model.

RELATED WORK

Video sharing is one of the most popular activities in the social media in which the familiar and trending videos are shared redundantly between cross applications among several users over a period which leads to not only increase in the data traffic across packet data network but also wastage of space and data resources of service providers and end users. Prior works were carried out based on the concepts of offloading the traffic to avoid backhaul congestion in the network and to reduce the user centric data response time with the introduction of caching the contents across various planes of the network and capitalizing on the predictive capabilities of the users' requests for proactively caching the contents. Konstatinos Poularakis et al [1] proposed an effective technique to improve the network performance in terms of energy consumption, throughput and user experienced delay, and at the same time make a better use of network resources such as wireless bandwidth and backhaul link capacity by considering a Heterogeneous Cellular Network with an optimization framework building on the combination of caching and multicast schemes. Chenchen Yang et al [2] focused on analyzing the performance improvement via caching technology in wireless heterogeneous networks by modeling the node locations (base stations, relays and users) of the three-tier HetNet (base stations-users, relay-users, users with caching ability-users) as mutually independent Poisson Point Processes which brings the contents very close to the user within the mobile network. Engin Zeydan et al [3] made use of the application of Big Data Analytics in mobile cellular networks by the introduction of a proactive caching architecture for 5G wireless networks to process huge amount of available data on a big data platform and leveraging machine learning tools for content popularity prediction to achieve backhaul offloading. Georgios Paschos et al [4] pointed out several technical misconceptions about insufficiency of static popularity models, tracking popularity variations, wired and wireless caching techniques and stakeholders' analysis of wireless caching for 5G networks. Wei Han et al [5] have emphasized to compare caching at physical and core network layers and to achieve the spectral efficiency gain over conventional caching schemes. Xiaofei Wang et al [6] proposed a new cooperative caching cell policy to reduce the duplicate traffic load within the MNO, which mostly focuses on content popularity prediction based on prefix-tree aggregation. Zhongxing Ming et al [7] focused on improving the performance of caching, save bandwidth cost for the mobile operators and to reduce the latency for the end users. The authors have proposed a caching strategy in two algorithms, offline greedy algorithm and Incan algorithm where the eNodeBs can use the information from the in-network router's caches to improve caching performance. In offline greedy algorithm, each time when content c_i which is not currently cached arrives, calculates its unit benefit value. If c_i has a higher unit benefit value than the object with the minimum unit benefit value that is currently stored, then replace the content with c_i. Incan algorithm uses a strategy where if the content c_i is not available in eNodeB, then its existence is verified in the in-network cache, if it is present, then it is pushed to eNodeB by following the Least Recently Used cache replacement strategy. Ejder Bastug et al [8] exploited the predictive capabilities to minimize the peak load of cellular networks by proactively pre-caching desired information to selected users before they request it. Xiaofie Wang et al [9] focused on the techniques related to caching in current mobile networks and proposed a novel edge caching scheme based on the concepts of content-centric networking and device to device communication where the requested content can be retrieved from the caching store of any device. Emir Halepovic et al [10] emphasized on improving the detection of the redundant traffic by chunk level matching and expansion while reducing the processing time through skipping the control packets like ACK, headers traversing in the traffic. The research works reported so far mainly focused on bringing the contents near to the edge of the network by using caching at various planes of MNO's infrastructure, but it is inevitable to handle the transmission of identical data between users from various devices.

PROPOSED SIMULATION MODEL FOR OPTIMAL TRAFFIC FLOW IN SOCIAL NETWORKS

An effective simulation model based on pre-existence of video frames extracted from the video contents is proposed to minimize the amount of traffic flow due to sharing of identical data among single or group of users through social media. The conceptual video sharing scenario considered for study is shown in the block diagram as given in Figure 1 and the roles of the stakeholders, Users and Server Application are detailed as follows:

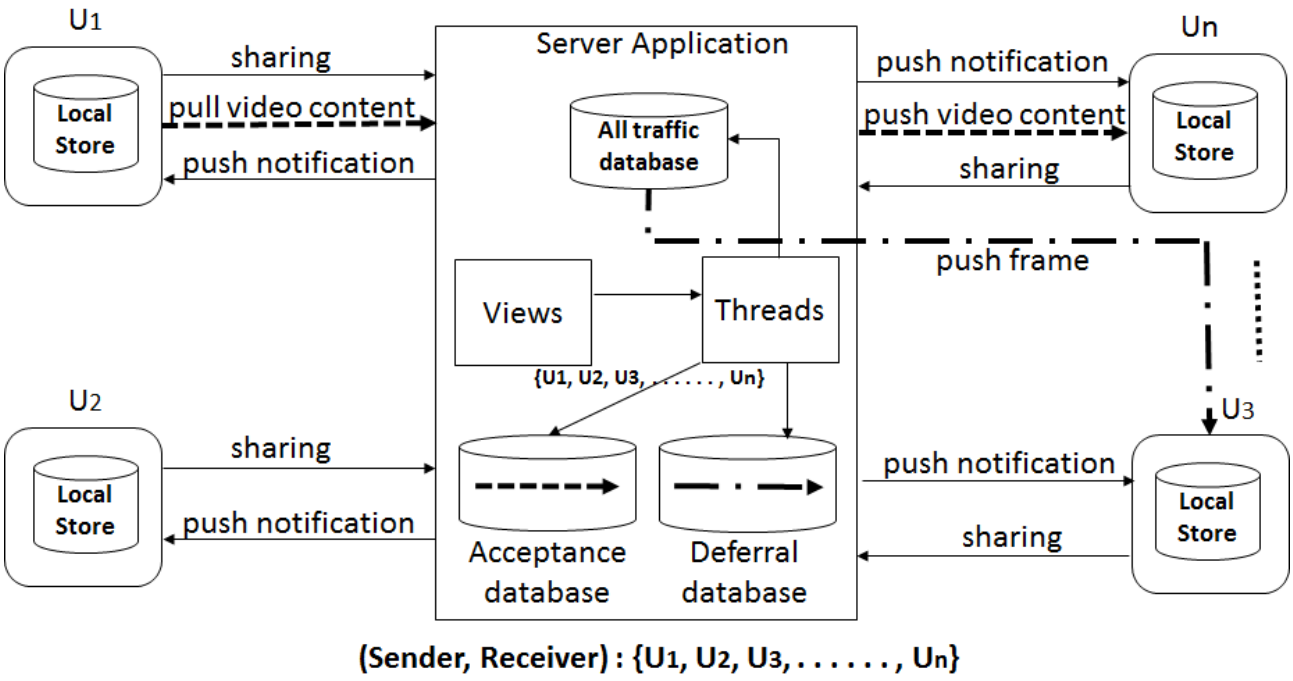

Figure 1. Conceptual Video Sharing Scenario

Users: {u_1, u_2, ..., u_n}

In the social media environment, a user can act as a sender of any valid content or a receiver of the same and vice versa. Users are the most important stakeholders of the proposed model, where a group of registered and mutually agreed users are involved in sharing the contents to other group of users. Every user has a local store where all the multimedia contents, which have been created locally or received from external social media environment are stored as follows:

{Category, Video_ID, Location, Size, Play Duration, Timestamp, Video Frame}

Where Category is of Boolean type (0 – Own Content, 1 – Received Content), Video_ID is a unique identifier comprises of the 'MAC' address of the device followed by '_' and a sequence number prefixed with 'vid', Timestamp shall be the time at which video has been created locally or received from external sources and Video Frame is the first frame of the actual Video Content of given Size and Play Duration and stored in the Location. To summarize, the users' local store is a database of the contents owned or received by any user, which acts as a repository for sharing the video contents through social media. Every user, as an originator of any video sharing transaction, extracts the video frame from the local store and creates a data frame, comprising Video_ID, Receiver(s) credentials, Video Frame and the Timestamp and forwards it to the server application for further processing.

Server Application

The server application is the predominant stakeholder of the proposed model, which serves all the registered users, aiding them in handling the video sharing process and manages the optimal flow of contents over a network. Any video sharing transaction initiated by the user is handled by pushing notifications to the receivers with the support for the comparison of video frames to take appropriate decisions. Every incoming data frame is prefixed with the sender's credential and inserted in the "all traffic database" from which the server segregates the traffic and creates separate Views based on the receiver's credentials. The server application accounts for the accepted or deferred transactions based on the pre-existence of the video frame in the View generated from the local store of the server and the respective receiver in case of deferred decision. Both "acceptance" and "deferral" databases store the data usage of the user (receiver perspective) as well as the total memory consumed in the device storage before and after the decision of the server application.

The first frame of the video content is used for comparison while initiation of sharing the content by push operation at the sender end. A server application, which is accessible for all registered users stores all the first frames of respective video contents along with the sender and receiver credentials. In the social media environment, each smart device plays the role of both sender and receiver. The server application maintains separate list of senders and receivers to do further statistical analysis of transactions i.e., pertaining to number of transactions made by every registered user and to find the percentage increase in the data as well as memory usage.

The functionalities of the proposed model are divided into four layers as shown in Figure 2, which comprises of Generation of video sharing traffic layer, Traffic segregation layer, Video frame validation and Push Notification layer and Identical Content sharing Analysis layer.

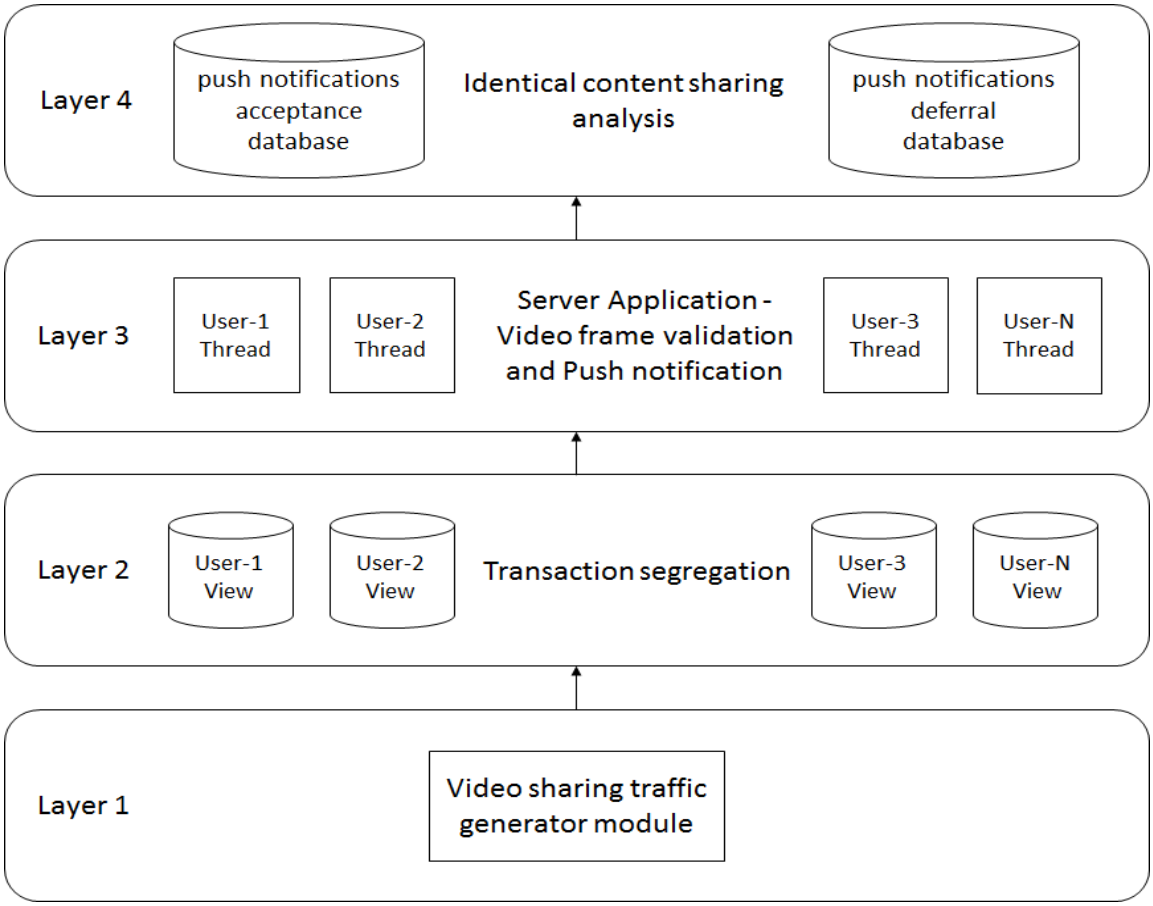

Figure 2. Various layers of proposed simulation model

Layer–1 is the traffic generation module, where video sharing traffic are populated randomly. The traffic may be tricky where Sender-A may share a video content to Receiver-A and at the same time Receiver-A shall share some other or same video content to Sender-A, since traffic is unpredictable, and every sender will be a receiver and vice versa in the social media environment. In addition to the video's unique id, the metadata for the video contents comprising location of the video file in the respective user's $\{u_1 \ldots u_n\}$ device from where the sharing is initiated (sender perspective), mode of sharing i.e. single or group of users, size and duration of play of the video contents along with sender and receiver credentials have been maintained in the data store of this module. The traffic is generated continuously for every specific period and pushed to the server application one record each time sequentially. The server application creates a View from the pre-existing records that are filtered with respect to the receiver credential as shown in Layer–2. At this stage, the data store in each user View contains the sender credentials, the first frame of the video content being shared, size and duration of play of the content and the unique id for that content generated at Layer–1.

The server application in Layer–3 co-ordinates with the Views and Threads to validate the video frame and takes appropriate decision. The server application checks the user View for the pre-existence of the video frame which is currently shared. If the video frame exists, a separate thread of control is initiated by the View to notify the receiver about the identical content and move the existing video content to the current location using the timestamp. The data frame is recorded in the "deferral database" by updating the details of the identical video such as sender credentials, size, duration, new location in the user local store (receiver perspective) and the current timestamp. If the frame does not exist, the server application obtains the location of the video content from the sender and pulls the video and pushes the same to the receiver. The data frame is recorded in the "acceptance database" which represents a non-identical content accepted by the receiver. For every sharing request, the local store in the server application is getting updated. Layer–4 is all about the analysis of the simulated model where the results have been discussed elaborately.

IMPLEMENTATION

In real-time, there are several social networking applications and messenger services like Facebook, WhatsApp, Instagram, Twitter, LinkedIn etc., where the mutually agreed groups of users can share the multimedia contents. It has been observed that, there is a paradigm shift in social media from using the mobile phones for voice calling and text messaging to huge amount of real-time video sharing and transmitting other multimedia contents.

To be precise, the era of sending "Hello" in a text message is now dominated by sending video contents like Graphic Interchange Formats (GIFs) or other video contents having feelings which can vary from 5MB to even 30 MB or more in size. In this research, a video sharing scenario is simulated with Python as front end and MySQL database as backend taking into consideration of sharing among cross applications in social media. The reason behind considering the video sharing superior to other multimedia contents is, it is dominating the entire traffic through packet data network using smart devices.

The simulation is based on the following scenario: a group of users has been considered those are using different applications with video sharing capability for exchanging the content to their close circle. The mode of video transmission is also considered where any user can send a video to a single user or to a group of users for sharing the contents simultaneously. Sets of {15 users, 100 Videos} and {50 users, 200 videos} are used as a base for the simulation of data traffic and are easily scalable by setting the respective parameters without effecting the other modules used for simulation. The users are expected to be operating through heterogeneous MNOs geographically distributed among various Public Land Mobile Network (PLMN) areas.

The parameters used for the generation of traffic are detailed as follows:

A sender or receiver is represented using a string 'UserX' where X is a sequence number. The video files are selected randomly from the user's local store using Video_ID. The play duration, size of the video and size of the first frame of the video are also randomly generated in the range of values 20 to 150 seconds, 2.0 to 7.0 Megabytes and 60 to 150 Kilobytes respectively. The first frame of the selected video is captured and the system time is taken as the initiated timestamp of the transaction. All these parameters form the data frame of the transaction initiated by the user (sender's perspective). The proposed algorithm discussed in section 3 is implemented in Python with MySQL database as backend. The logical schema for various database tables and views implemented in the simulation model are given below:

Videos metadata

[Category, Video_ID, Location, Size, Duration, Video_Frame, Timestamp]

Users metadata

[User ID, MNO_ID, MNO_Name, PLMN_ID, PLMN_Area]

Video Sharing Traffic Database ("all traffic database" in server application)

[Sender, Video_ID, Size, Duration, Video_Frame, Receiver, Initiation_Timestamp]

User View (generated using Receiver credentials**)**

[Sender, Video_ID, Size, Duration, Video_Frame, Initiation_Timestamp]

push_notifications_accepted ("acceptance database" in server application)

[Sender, Video_ID, Receiver, Size, Duration, Video_Frame, Memory_Usage_Status, Data_Usage_Status, Relocation_Timestamp]

push_notifications_deferred ("deferral database" in server application)

[Sender, Video_ID, Receiver, Size, Duration, Video_Frame, Memory_Usage_Status, Data_Usage_Status, Relocation_Timestamp]

Layer–1: Generation of video sharing traffic

The packages MySQLdb, numpy, csv, random, datetime and time are imported in 'GenerateUserVideosharingTraffic.py' to generate the pseudo random traffic samples. The code snippets pertaining to choose a sender, video file, play duration, size, receiver, initiation timestamp are given below:

```
#Function for choosing a random sender
def selectRandomUser():
    query ='SELECT USER_ID FROM USERS_METADATA'
    cursor.execute(query)
    allUserNames = cursor.fetchall()
    listOfUserNames = []
    for userName in allUserNames:
        listOfUserNames.append(userName[0])
    #gets a sender as randomUser
    randomUser = random.choice(listOfUserNames)
```

```
#Function for choosing a random video
def selectRandomVideoFile():
    query = 'SELECT VIDEO_ID FROM VIDEOS_METADATA'
    cursor.execute(query)
    allVideoFiles = cursor.fetchall()
    listOfVideoFiles = []
    for videoFile in allVideoFiles:
        listOfVideoFiles.append(videoFile[0])
        randomVideoFile = random.choice(listOfVideoFiles)
    return(randomVideoFile)
```

```
#Function for choosing the play duration of the selected video
def selectRandomVideoFileDuration(randomVideoFile):
    query = "SELECT DURATION FROM VIDEOS_METADATA
              WHERE VIDEO_ID = '%s' "%(randomVideoFile)
    cursor.execute(query)
    randomVideoFileDuration = cursor.fetchone()
    return(randomVideoFileDuration[0])
```

```
#Function for choosing the size of the selected video and video
frame
def selectRandomVideoFileSize(randomVideoFile):
    query = "select video_size,video_first_frame_size from
videos_metadata
              where video_id = '%s' "%(randomVideoFile)
    cursor.execute(query)
    randomVideoFileandFrameSize = cursor.fetchone()
    return(randomVideoFileSize[0],randomFileandFrameSize[1]
```

```
#Function for choosing group of receivers list for a sender
def selectRandomReceiversList(randomUser):
   query = "SELECT USER_ID FROM USERS_METADATA WHERE
            USER_ID NOT LIKE '%s'"%(randomUser)
   cursor.execute(query)
   receiversUsersTuple = cursor.fetchall()
   rlist = []
   for receiver in receiversUsersTuple:
      rlist.append(receiver[0])
          nReceivers=random.randint(1,20)

selectRandomRList=(np.random.choice(rList,nReceivers,replace=False))
   count = len(selectRandomRList)
   selectRandomRNamesList = []
   x = 0
   while(x < count):
      selectRandomRNamesList.append(selectRandomRList[x])
      x+=1
   return count, selectRandomRNamesList
```

```
#Function for choosing the timestamp
def selectRandomTimeStamp():
start = datetime.datetime.now()
end = start + datetime.timedelta(numberOfDays)
randomTimeStamp = start + (end - start) * random.random()
return(randomTimeStamp)
```

The randomly generated parameters are passed to the video traffic generate method, i.e., to generateTraffic(). The code snippet pertaining to generation of the video sharing traffic is given below:

```
#Function for generating the video traffic
def generateTraffic(Sender, Video_ID, Receiver, Size, Duration,
                    Video_Frame, TimeStamp):
   fileExists = os.path.isfile(myCSVFileName)
   with open(myCSVFileName,'a',newline='') as CSVFile:
      csvHeader = ['SENDER','VIDEO_ID',' RECEIVER', 'SIZE',
                   'DURATION',  'VIDEO_FRAME',
                   'VIDEO_FRAME'_SIZE , 'TIMESTAMP']
      writeHeader = csv.DictWriter (CSVFile, fieldnames = csvHeader)
      if not fileExists:
         writeHeader.writeheader()
      CSVFile.write(Sender)
      …
      CSVFile.write(randomTimeStamp.strftime(myTimeFormat))
      CSVFile.write('\n')
```

Layer–2: Traffic segregation

For each sharing of video, the server application creates a View based on the records from its local store filtered using the Receiver credential as in the data frame received by it. The sample code snippet for the creation of user Views is given below:

```
def createReceiversView(viewName, receiverName):
   query = "CREATE OR REPLACE VIEW %s AS SELECT * FROM
            video_sharing_traffic_database WHERE RECEIVER = '%s'
            ORDER BY TIMESTAMP"%(viewName,receiverName)
   cursor.execute(query)
   db.commit()
   query = "select distinct receiver from video_sharing_traffic_database"
   cursor.execute(query)
   allReceivers = cursor.fetchall()
   receiverNameList = []
   for receiverNameTuple in allReceivers:
      receiverNameList.append(receiverNameTuple[0])
   length = len(receiverNameList)
   for receiverName in receiverNameList:
      viewName = receiverName+'_Transactions'
      createReceiversView(viewName, receiverName)
```

Initially, server local store does not contain any history of transactions, but gradually the incoming data frames are populated as and when the decision is taken by the application either the push notification is accepted or deferred. An empty result set of the user View does not mean that the receiver does not have the same video content, since the user shall be registered later to the server application. In this scenario, a separate thread of control is initiated by the server for verifying the existence of the video frame directly at the receiver end, if the video frame exists it indicates that the receiver has already obtained the video through some unregistered user and accordingly, the transaction is recorded in the deferral database. If the frame does not exist in the receiver local store, the transaction is recorded in the acceptance database and subsequently in either case an entry is made in to the server local store. In Python, the csv.DictReader() method is used to differentiate the comma separated values in each row with the header of the CSV file and accordingly the values are inserted in the server local store. The code snippet pertaining to the insertion of the transactions into the server local store is given below:

```
#record the incoming traffic in video sharing traffic database[all traffic db]
record = [ ]
with open(myCSVFilename) as csvFile:
   reader = csv.DictReader(csvFile)
for line in reader:
   record.append(line)
for i in range(0,myCSVFileRowCount-1):
   Col1 = record[i]['SENDER']
   ….
   timeStamp = datetime. datetime.strptime(Col7,myTimeFormat)
   query = "insert into video_sharing_traffic_database
            (sender, video_id, size, duration, receiver,
             video_frame, video_frame_size, timestamp) VALUES
             ('%s', '%s', '%s', '%s', '%s', '%s', '%s')" %(col1,….,timestamp)
    cursor.execute(query)
```

Layer–3: Server Application - Video frame validation and Push notification

In Layer–2, it is focused to populate the server local store from the scratch for each initiation of sharing of video content initially or while a new user is registered, and sharing is initiated to that user. In Layer–3, a separate thread of control is executed by the server application for each data frame in the user View after transactions have been segregated. If any one of the video frame in the View matches with the one in the current data frame while comparing, the corresponding receiver should have the same video content which is being shared now and the server application defers the push notification. The video frame is stored in the local store as stream of bytes i.e., as BLOB object and the same will be retrieved using io.ByteIO() method.

The sample code snippets for capturing the first frame from the video content and comparing the video frames are given below:

```
#Capture a frame from a video
import cv2
def captureImagesFromVideo(folderPath,fileName):
    frameRetrieved = True
    captureVideoFrame = cv2.VideoCapture(folderPath+"\\"+(fileName))
    count = 1
    # A break statement before incrementing the count, captures first frame
    while frameRetrieved:
        frameRetrieved, image = captureVideoFrame.read()
        cv2.imwrite(folderPath+"\\Frame%d.jpg" % count,image)
        count+=1
```

```
def CheckFramesInUserView(inFrame,userViewFrames,numberOfRows):
    inFrame.show()
    flag = 0
    count = 1
    i = 0
    while(count <= numberOfRows):
        frame = io.BytesIO(userViewFrames[i][0])
        print('Verifying the Frame No: '+str(i+1)+'  In User View')
        userViewImage = PIL.Image.open(frame)
        userViewImage.show()
        if(inFrame == userViewImage):
            flag = 1
            print('Frame number  '+str(i+1) +' is matched ')
            break
        count+=1
        i+=1
        return flag
    if flag == 1:
        #Acceptance procedure
    else:
        #deferral procedure
```

Layer–4: Analysis of identical content sharing

To verify and validate the proposed model, the data usage and memory information stored in the push notifications accepted and deferred data stores have been analyzed prior and after application of the proposed model. The attributes considered in the execution of the test cases are given in Table 1.

Table 1. Generating Video Sharing Traffic – Simulation Attributes

Attribute	Description	Value(s)
U	Number of registered users	15 and 50
V	Number of videos for sharing	100 and 200
D	Play duration of a video	20 to 150 seconds
S	Size of a video	2.0 to 7.0 MB
F	Size of the first frame of a video	60 to 150 KB
M	Mode of Video Sharing	Single or Group
P	Simulation Period	30 days
I	Number of initiations for sharing of video	200 to 2500

Twelve test cases have been considered for execution based on the attribute values U = 15, V=100, P = 30 days, S, D, F and I ranging between the values as given in Table 1. The total transactions generated, push notifications accepted and deferred after simulation are given in Table 2.

Table 2. Total Transactions and Accepted vs Deferred Percentage of Push Notifications

Test Case	No. of Initiations	Total transactions generated	No. of push notifications			
			Accepted	Acceptance Percentage	Deferred	Deferral Percentage
1	200	806	620	76.92	186	23.08
2	400	1643	1040	63.30	603	36.70
3	600	2388	1180	49.41	1208	50.59
4	800	3220	1325	41.15	1895	58.85
5	1000	3954	1382	34.95	2572	65.05
6	1200	4789	1449	30.26	3340	69.74
7	1400	5461	1464	26.81	3997	73.19
8	1600	6354	1478	23.26	4876	76.74
9	1800	7135	1483	20.78	5652	79.22
10	2000	7984	1488	18.64	6496	81.36
11	2200	8826	1496	16.95	7330	83.05
12	2500	10073	1500	14.89	8537	84.75

Table 3. Saving of Data Usage by the proposed model

Senders	Total Video Push Initiations	Data Usage in existing scenario (MB)	Accepted Video Push Initiations	Data Usage by the proposed model (MB)	Deferred Video Push Initiations	Data Usage Saved (MB)	Data Usage Saved (%)
200	2057	8760	1819	7780	238	980	11
400	4165	18761	3500	15584	665	3177	17
600	6223	28120	4641	21027	1582	7093	25
800	8203	36720	5623	24985	2580	11735	32
1000	10562	46744	6482	28647	4080	18097	39
1200	12464	55063	7157	31705	5307	23358	42
1400	14731	63820	7784	34235	6947	29586	46
1600	17069	75901	8140	36188	8929	39713	52
2000	20706	91176	8740	38561	11966	52616	58
2500	26553	116821	9322	40850	17231	75971	65

From the results shown in Table 2, it is inferred that out of the randomly generated traffic where the number of video sharing initiations vary from 200 to 2500, the percentage of push notifications accepted is inversely proportional to the number of total transactions generated whereas the percentage of push notifications deferred is directly proportional, which represents that when more video contents are shared, the probability of having identical contents increases in a progressive fashion. At one point of time during the simulation i.e., after test case 12, where there are no newer videos available to be shared, the push notification accepted percentage remains zero and all the incoming notifications will be deferred resulting to avoidance of identical contents.

A second set of test cases are executed by increasing the number of registered users and videos for sharing with U = 50, V = 200, P = 30 days, S, D, F and I as per the range of values specified in Table 1. The results obtained are shown in Table 3.

In the context of social media environment, where any sender at a point of time can be a receiver and vice versa, the statistical analysis with respect to the amount of reduction in the redundant data traffic at the network layer and the amount of data usage saved for individual sender / receiver are derived and few samples of the sender / receiver who initiated the highest number of transactions and received highest number of notifications in each test case are shown in the Tables 4 and 5 respectively.

Table 4. Data Usage Status – Sender's perspective

No. of Initiations	Sender	Total Video Push Initiations	Existing Scenario	Proposed Model				
			Data Usage Required (MB)	Accepted Video Push Initiations	Data Usage Consumed (MB)	Deferred Video Push Initiations	Data Usage Saved (MB)	Data Usage Saved (%)
200	User42	102	503	93	477	9	26	5
400	User13	182	733	151	617	31	116	16
600	User3	249	1118	204	907	45	211	19
800	User41	274	1086	190	785	84	302	28
1000	User7	297	1499	196	981	101	519	35
1200	User19	431	1661	238	922	193	739	44
1400	User30	417	1874	212	992	205	882	47
1600	User4	523	2375	295	1349	228	1026	43
2000	User16	552	2342	238	1142	314	1199	51
2500	User31	708	3020	259	1103	449	1917	63

Table 5. Data Usage Status – Receiver's perspective

No. of Initiations	Receiver	Total Video Push Initiations	Existing Scenario	Proposed Model				
			Data Usage Required (MB)	Accepted Video Push Initiations	Data Usage Consumed (MB)	Deferred Video Push Initiations	Data Usage Saved (MB)	Data Usage Saved (%)
200	User26	54	223	46	189	8	34	15
400	User17	104	472	82	362	22	110	23
600	User42	144	661	99	453	45	209	32
800	User18	186	857	128	584	58	273	32
1000	User48	236	1062	139	612	97	450	42
1200	User16	287	1234	158	689	129	546	44
1400	User23	324	1379	155	663	169	716	52
1600	User26	380	1648	168	743	212	906	55
2000	User26	451	1998	170	747	281	1252	63
2500	User22	573	2577	189	848	384	1729	67

The overhead for transmitting the first frame of the videos is also analyzed and it has been observed that an additional 2.4 percentage of data usage is consumed for the transmission of the first frames of the videos during User31's initiated video sharing for 2500 times as given in Table 4 and hence the amount of saved data usage given in the last column (i.e., 63 percent in this case) does not portrait the overhead due to the transmission of video frames instead of actual video contents. It is inferred that the overhead for the transmission of the first frame of the videos when compared to the actual video contents is negligible when number of initiations are more which is an ideal scenario for video sharing environment. From the simulation statistics observed, cluster of user categories like light, moderate and elite group of users can also be inferred by considering a sender with respect to minimum, average and maximum number of initiations among all the registered users involved in video sharing process. These clusters can provide support to the MNOs to identify the privileged customer for providing them special tariffs and value-added services.

CONCLUSION

The proposed simulation model is completely implemented using Python considering the social media applications those are capable of handling multimedia contents. Currently, if the same video content is shared from one application to another application i.e., sharing between cross applications, both sender and receiver experience data usage equal to the size of the video content. This issue is addressed in the simulation model to minimize the mobile data usage at both ends if the receiving device has the same content already. In this scenario, the actual content is not shared from the sender device instead it will be copied to the current location of the receiver device from its local store. This approach not only minimizes the usage of mobile data due to sharing of identical contents but also reduces the network traffic invariably, which results in the optimal utilization of network bandwidth and reduction of the operating costs of the service provider. The simulated results show that among the content routed through the Internet, due to the popularity of trending video sharing, nearly fifteen percent of the videos are shared multiple times across various users over a period and the percentage of deferred push notifications increases to nearly eighty percent which avoids a huge congestion in the network layer. Further the results indicate that the increase in the number of times of initiation of video sharing i.e., from 200 times to 2500 times, there will be saving of data usage in the range fifteen to sixty-seven percent with a significant reduction in the identical data traffic by the implementation of the proposed model irrespective of the overhead associated with the transmission of video frames instead of the actual video contents. The more the frequency of the video sharing causes increase in the percentage of redundant data transmission and hence the percentage of deferred push notifications, which saves the routing costs involved with various stakeholders like sender, receiver and the service provider. By tackling the bottleneck of redundant data transmission, the enhancements that are essential for the improvement of performance metrics of the network are addressed, which will aid the next generation mobile networking standards.

REFERENCES

[1] Konstatinos Poularakis, George Isofidis, Vasilis Sourlas, Leandros Tassiulas, 2016, "Exploiting Caching and Multicast for 5G Wireless Networks", IEEE Transactions on Wireless Communications, Vol.15, No. 4, pp. 2995-3007.

[2] Chenchen Yang, Yao Yao, Zhiyong Chen, Bin Xia, 2016, "Analysis on Cache-Enabled Wireless Heterogeneous Networks", IEEE Transactions on Wireless Communications, Vol. 15, No. 1, pp. 131-145.

[3] Engin Zeydan, Ejder Bastug, Mehdi Bennis, Manhal Abdel Kader, Ilyas Alper Karatepe, Ahmet Salih Er, Merouane Debbah, September 2016, "Big Data Caching for Networking: Moving from Cloud to Edge", IEEE Communications Magazine, pp. 36-42.

[4] Georgios Paschos, Ejder Bastug, Ingmar Land, Giuseppe, Merouane Debbah, August 2016, "Wireless Caching: Technical Misconceptions and Business Barriers", IEEE Communications Magazine, pp. 16-22.

[5] Wei Han, An Liu, Vincent K.N. Lau, August 2016, "PHY-Caching in 5G Wireless Networks: Design and Analysis", IEEE Communications Magazine, pp. 30-36.

[6] Xiaofei Wang, Xiuhua Li, Victor C. M. Leung, Panos Nasiopoulos, 2015, "A Framework of Cooperative Cell Caching for the Future Mobile Networks", Proceedings of the 48th Hawaii International Conference on System Sciences, IEEE Computer Society, pp. 5404-5413.

[7] Zhongxing Ming, Mingwei Xu, Dan Wang, 2014, "InCan: In-network cache assisted eNodeB caching mechanism in 4G LTE Networks", Computer Networks Elsevier Journal, Vol 75, pp. 367-380.

[8] Ejder Bastug, Mehdi Bennis, Merouane Debbah, August 2014, "Living on the Edge: The Role of Proactive Caching in 5G Wireless Networks", IEEE Communications Magazine, pp. 82-89.

[9] Xiaofie Wang, Tarik Taleb, Adlen Ksentini, Victor C.M. Leung, February 2014, "Cache in the Air: Exploiting Content Caching and Delivery Techniques for 5G Systems", IEEE Communications Magazine, pp. 131-139.

[10] Emir Halepovic, Carey Williamson, Majid Ghaderi, 2011, "Enhancing redundant network traffic elimination", Computer Networks Elsevier Journal, Vol 56, pp.795-809.