Cloud Based Simulation Model for Traffic Flow Optimization in Online Social Networks

Nagaraju Baydeti
Department of Computer Science and Engineering
National Institute of Technology Nagaland
Chumukedima, Dimapur 797 103, Nagaland
e-mail: baydetinagaraju@nitnagaland.ac.in

Ramachandran Veilumuthu
Department of Information Science and Technology
College of Engineering Guindy, Anna University
Chennai 600 025, India
e-mail: rama5864@auist.net

Mariappan Vaithilingam
Software Development Manager
Amazon, Bengaluru 560 103, India
e-mail: dr.v.m@ieee.org

***Abstract* — The modern online social networks consist of many interconnected services, such as business, private, public, entertainment and social media, which are collectively represented as Social Networking Services. The size of the social network grows exponentially due to increase in social media services and their real use in the above stated applications. The main aim of this paper is to develop a cloud-based simulation model that provides video sharing, content storage and appropriate decision-making services on demand over the Internet with an objective of reducing the redundant traffic flow amidst the applications and hence to utilize the network bandwidth effectively in the online social networks. The computational engine, which is included in the proposed cloud model will inherently analyse the traffic flow due to video data transmission across social media applications. The service providers can deploy their applications without any limitation in a cloud environment and users can share information using the rich deployed applications from anywhere on demand basis or on the "push and pull" basis. The deploying services minimize the risk of redundant traffic, which not only reduce the consumption of network bandwidth in the perspective of service providers but also lower the cost of usage of mobile data in the perspective of end users. The cloud-based simulation model provides reliable services using virtualized compute and storage technologies. The performance measures such as network bandwidth usage, mobile data usage and time consumption are estimated before and after sharing of video contents among different social media applications, which are most popular in the online social networks.**

Keywords - Online Social Networks; Redundant Data Traffic; Video Sharing; Django Web Framework; Google Cloud Platform.

I. INTRODUCTION

During the late 90's when very few social networking applications came in to existence, the primary motive was to upload the profiles and get connected socially by making friends and the content sharing was predominantly text based. During those initial periods, the accessibility of such applications was through personal computers and laptops via World Wide Web. Over the period, due to the advancements in networking technologies, higher data rates provided by the Internet Service Providers (ISPs) have paved a path to expansion in the social networking services. Even though the concept of smart phones came in to existence in the early 2000, the vide usage of smart phones have increased drastically due to the introduction of relatively low cost smart phones over the last decade and now smart devices have multi-core processing units with gigabytes of memory, high resolution digital cameras and affordable applications enabled the users to create their own multimedia content for sharing in the social media and other applications in the social networks. In addition to the affordability, the immense and rapid developments in the enhancement of smart phone operating systems, especially with Android, iOS etc., made social networking applications available to every common man.

At present, there is a paradigm shift in the social networking applications from profile updates and text content sharing to voluminous exchange of multimedia contents. Currently, video sharing through exhaustive list of social network applications is the most popular scenario in the field of communication where the trending videos are shared among several users simultaneously over a period, which leads to increase in the data traffic across packet data network. Various types of data have been utilized in social media applications to share the same in different formats in a heterogeneous service providers' environment. The social media applications have been developed and deployed in different platforms for wide usage by different service providers keeping certain things in common including specific or customized services. 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, it becomes a bottleneck for the service providers with respect to the data rates. 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. 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. In this scenario, redundant data elimination will aid all the service providers to accomplish the specific service level agreement targets set to all their respective customers.

Considering the immense raise in the number of social media applications, connected IP based devices and the identical traffic flow, a generalized cloud framework is proposed for regulating the capabilities of the Internet layer to optimize the flow of data traffic especially while sharing of video contents. It is observed that when identical video is being shared across several users within the same application, for example if a trending or own video is shared among multiple groups, the quantum of data usage and the replica of packets being transmitted is handled properly, as the server identifies the contents are identical as it is being initiated or received again and again by the end users. In case of video sharing across different social networking applications, the identical video contents are routed as per the service providers' infrastructure, the quantum of data usage and the replica of redundant identical packets being transmitted in multiple routes might create congestion in the packet data network.

II. 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. Chenchen Yang et al [1] focused on analysing the performance improvement via caching technology in wireless heterogeneous networks by modelling the node locations (base stations, relays and users) of the three-tier HetNet (base stations-users, relay-users, users with caching ability) as mutually independent Poisson Point Processes which brings the contents very close to the user within the mobile network. Engin Zeydan et al [2] 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. Xiaofei Wang et al [3] proposed a new cooperative caching cell policy to reduce the duplicate traffic load within the Mobile Network Operator, which mostly focuses on content popularity prediction based on prefix-tree aggregation. Zhongxing Ming et al [4] 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. Ejder Bastug et al [5] 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 [6] 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. 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 across different social network applications. Myoungjin Kim et al [7] presented a novel social media model integrated with the cloud computing environment to provide elastic computing resources for processing and storing big social media data and platforms for developing social network services.

Google Analytics plays a tremendous role in analysing the social media applications like Facebook, Twitter etc., by keep tracking the associated Websites using a unique tracking code. The "Dimensions" and "Metrics" are the most important building blocks of Google Analytics [10]. Dimensions are the attributes of the data such as, from where the traffic is initiated, i.e., the Website which is currently being analysed or the social media application which is being evaluated. Metrics are the quantitative measurements for the dimensions. Google Analytics assists in the social media analytics to monitor and analyse the social media data, allows to make decisions based on the data and extracts useful information and patterns.

Neil T. Spring et al. [12] proposed a protocol independent cache framework in which the packet granular data was considered to find the redundant traffic in the network. This technique identifies the repeated byte ranges between the packets to avoid the retransmission of the identical data. Wataru Yokota et al. [13] have made a comprehensive study on Traffic Reduction (TR) node which reduces the TCP redundant traffic that is repeated in short time using packet caches. Reduction of identical traffic between two TR-nodes is achieved by caching the contents, where the upstream TR node reduces the TCP data size by

encoding when its cache contains the same data as received and the downstream TR-node restores the encoded data to the original data using its cache. The main focus is to reduce the identical traffic with the usage of smaller caches. The research works reported so far focuses to bring the contents near to the edge of the network of the service provider for reducing the backhaul congestion and also to reduce the redundant traffic especially for the video on demand services. Inspite of tremendous publications reported so far in the area of Internet traffic flow analysis, it is hard to find publications related to redundant data traffic flow minimization across online social networks. The tremendous increase of mobile data traffic across Internet layer is due to the voluminous video-on-demand and video sharing services offered by various social network applications, where the familiar and trending videos are viewed or shared among several users over a period. The popularity gained by few videos over a period of time results in the increase of sharing identical contents which is commonly observed and handling of this redundant data traffic especially across social media applications is addressed in this work.

Cloud environment provides anytime anywhere solutions for data sharing and integration issues. The scalability is the major issue in the current online social networks. The social network services are used by many concurrent users at a time. The system must be designed to handle operations by the clients and providers without any hurdle to exchange data due to heterogeneous nature. The cloud environment provides the inherent dynamic scalability for the operations of social networks. Cloud computing does not require global standard architectures as comparable to grid computing, and it does not necessarily need a standard, open, general-purpose protocol. Furthermore, cloud computing supports interfaces that are syntactically simple, semantically restricted and high-level. These features of interfaces are underlying factors for a rapid adoption of cloud computing services in the social networking applications.

An effective cloud-based simulation model based on pre-existence of video frames extracted from the video contents is proposed in this paper to minimize the amount of traffic flow due to sharing of identical data among single or group of users through social media. To make its implementation more general and scalable for real time analysis, cloud services are introduced in different layers in accordance with MVT pattern of Django Web Framework [9] and deployed in Google Cloud Platform. Analogous to the Google Analytics, the basic building blocks of the proposed cloud-based model is to analyse the various dimensions and metrics such as, network bandwidth, time consumption and the data usage statistics, highest number of initiations made by a sender, highest number of notifications to the receiver, highest number of video sharing incoming requests and download and upload speeds. With the increase in the usage of Social Media, analysis of the usage is also made by providing graphical reports like percentage saving of mobile data and percentage of accepted versus deferred notifications, etc.

III. PROPOSED CLOUD BASED SIMULATION MODEL – A LAYERED APPROACH

The cloud-based simulation of video sharing scenario in social networks is described in layers as shown in Figure 1.

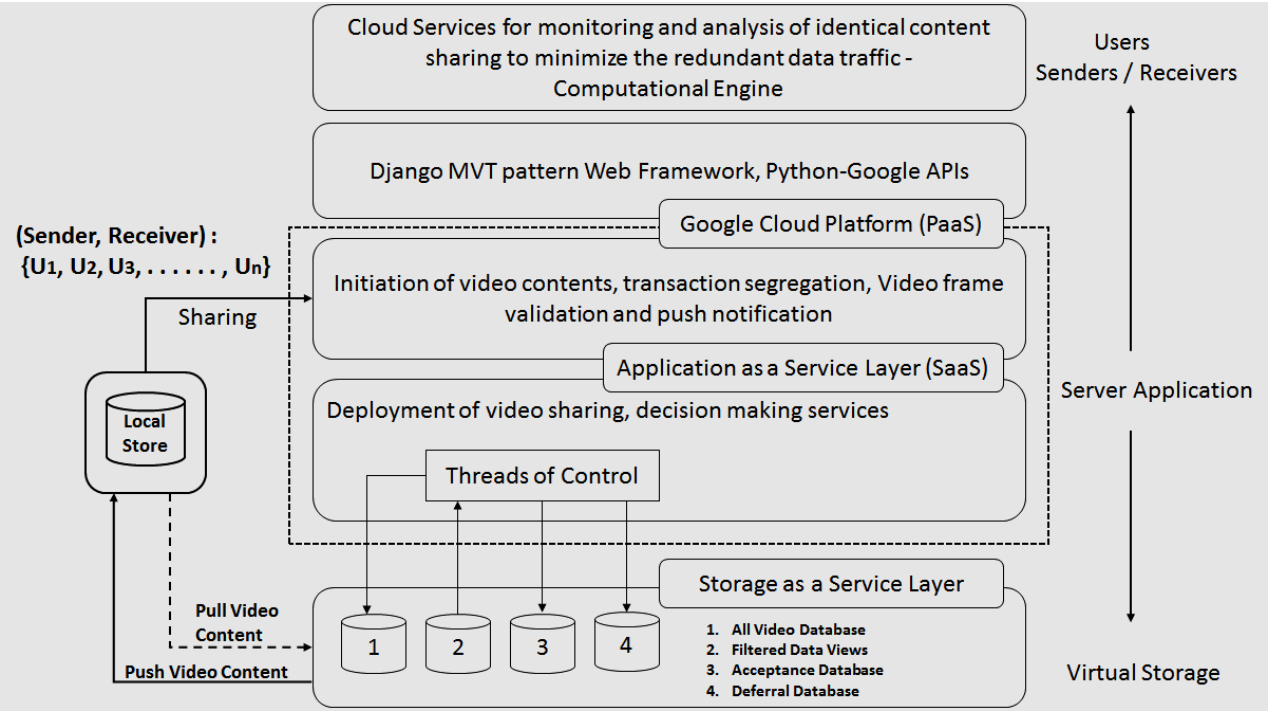


Figure 1. Conceptual Video Sharing Scenario – Cloud Based Simulation

The main layers [8] of the proposed model are the "Platform as a Service" (PaaS) layer, which is Google Cloud Platform, Software or Application as a Service (SaaS) layer where all the proposed services are deployed and Storage as a Service layer where all the databases have been maintained. In social media applications, the clients are the senders or receivers from or to whom the videos have been shared and generally they are smart devices which initiate video sharing services. In general, the Users and Server Application are the stakeholders of the proposed model with a Computational Engine to perform the statistical analysis of traffic flow. The roles of the stakeholders and the different layers of the proposed cloud framework are detailed in the following sections:

A. End Users or Smart devices: $\{u_1, u_2, \ldots\ldots, u_n\}$

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 using the metadata: {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.

B. 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 Filtered Data 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 Filtered Data 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.

C. Infrastructure as a Service (IaaS)

The cloud infrastructure layer provides the fundamental resources needed to share upper level platforms and services. The "Storage as a Service" layer (Model perspective) and the physical resources needed for the "Computational Engine" and for communication among services along with core server application (View perspective) form the basis for delivering Infrastructure as a Service (IaaS). All the services are developed in Python and are more interoperable due to XMLised messages for communication. The service providers are free to design their "apps" directly on top of this layer, skipping the platform layer. This results in increased freedom and flexibility, since providers can opt to use an existing platform that matches the individual system, or even implement their own platform for specific cases. This approach is used to transfer social networking services to a cloud to reduce infrastructure investments.

D. Platform as a Service (PaaS)

Google App Engine is used as PaaS in the proposed model that provides a conducive environment to implement the Web services. Google Cloud provides a set of APIs to aid the interaction between cloud components and end user applications, to enhance scalability, and ease deployment and management. The Django Web Framework provides an environment to deploy the services developed in accordance with Model-View-Template (MVT) pattern and the "google-cloud" API for Python provides interfaces for interacting with Google App Engine, which is configured as Python based cloud runtime environment. Google App Engine provides useful features to ease development and reduce the development time, including automatic scaling and load balancing, as well as integration with other services.

E. Software as a Service (SaaS)

Software as a Service layer is for end users i.e., for the application developers. The services in this layer are typically accessed through Web portals using Templates. The end users will feel the easiness of the proposed cloud computing model and shall appreciate because it alleviates the need to provide support for hardware to run applications and services, as well as eliminate the need for local installation and configuration. The clients can compute their

required services from their terminals. The social networking services provided with this model are normally referred to as SaaS and implemented in Views of Django Web Framework. The greatest advantage of providing services in this manner is that the clients are unable to access the source code. The service providers can roll out new features and updates without disturbing the clients, if the system is backward compatible with existing data.

F. Software as a Service (SaaS)

The content providers of the various social media applications, which have been developed by various social network service providers have stored the data in different formats under heterogeneous environment. The server application in the cloud environment has the control to manage the databases maintained in the Storage as Service layer. With the App Engine, the social networking service providers can write their application code, test it on their local machines and then deploy on cloud environment. Google App Engine provides scalable technologies like Google File System, BigTable and Cloud SQL for building applications. Cloud SQL is a scalable distributed database system for large distributed data intensive applications where the second-generation instances support MySQL in cloud. Google Cloud also provides a well-established secured TCP tunnel via standard TCP port between the local development environment and the Cloud SQL instance. Cloud storage services must meet several requirements, such as high availability, adequate performance, replication and consistency.

IV. CLOUD BASED SIMULATION MODEL FOR TRAFFIC FLOW OPTIMIZATION – IMPLEMENTATION

The Web services developed for monitoring and analysing the traffic flow in the social networks due to sharing of redundant and trending video contents have been deployed using Google App Engine in the Google Cloud. The App Engine is used as a Platform as a Service (PaaS), where the server application is hosted using Django Web Framework adopting MVT pattern. The Google App Engine is dynamically scalable based on the traffic involved in the server application. The "Users" are the primary stakeholders of the model where a group of mutually agreed users are involved in video sharing process through social media applications. In this model, the "Users" are considered to be spatially located at various geographical locations and registered under different Mobile Network Operators (MNOs) or Internet Service Providers (ISPs).

Each individual component in the Application as a Service layer is a Web service or a Module that encapsulates a set of related functions (or data). The services and modules have been described as scripts as well as they are structured with technical requirements, related constraints and mechanisms for access or response. The descriptions are in the form in which their syntax, structure and semantics are widely accessible, and they conform to the standards and specifications of Django Web Framework. This layer closely associates with the Storage as a Service and Platform as a Service layers to access the Cloud SQL and the Python scripts implemented in the View component of Django framework, while the complete set of actions has been implemented in Model-View-Template (MVT) pattern and creates the communication artifacts.

The traffic generation service defined in the Application or Software as a Service Layer (SaaS) is initiated to populate video sharing traffic randomly at specified time intervals. In the social media environment, every user shall act as a sender or a receiver and the traffic is generated in unpredictable manner due to random pushing or pulling of information i.e., sharing of video contents. In principle, the metadata of all the shared video contents are maintained in the Cloud SQL. Each video has been assigned a unique id using the mac address of the machine from where it is being shared, followed by the symbol "_" and a sequence number prefixed with a string "vid". In addition to this unique id, the location of the video file in the respective user's {u1…..un} 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 are also stored. The traffic is initiated continuously for every specific period and pushed to the server application one record each time sequentially. The server application creates a Filter Data View from the pre-existing records that are filtered with respect to the receiver credential. At this stage, the data store in each Filter Data 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.

The server application co-ordinates with the "Filtered Data Views" and Threads to validate the video frame and takes appropriate decision. The server application checks the "Filtered Data 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 "Filtered Data 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 ("All Video Database") in the server application is getting updated. The databases pertaining to "All Traffic", "Acceptance" and "Deferral" constitute the virtual storage component of the cloud. The "Computational Engine" layer performs the statistical analysis based on the traffic flow simulation and the results have been generated. In the proposed model, all the services are stored in the centralized location. The server in the cloud environment has the control and can manage not only the services stored in the "Application as a Service" layer but all

also all the data stores maintained in the "Storage as a Service" layer. The "Computational Engine" layer is provided on top of all layers to monitor the traffic flow and to minimize the redundant traffic in accordance with the statistical analysis.

A. Creating, Uploading and Registering Django Web Services – Cloud Services

The Python based cloud interfaces provided by the "google-cloud" client API as well as the shell SDK are location independent and can be accessed by well-established interfaces like Django Web Services framework and Internet browser. The Google App Engine is used for registering, uploading and accessing the video sharing services in the proposed cloud computing model for minimization of redundant traffic flow among social media applications. The App Engine Software Development Kit (SDK) for Python is used to create and to link the services to the cloud. The cloud applications need a configuration file i.e., "app.yaml" to deploy and run the application. The video sharing services configured using App Engine are easy to build, easy to maintain, and easy to scale as traffic and data storage needs grow. The applications have been uploaded for ready to serve to the end users.

B. Simulation – Experimental Setup

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. Ten different Service Providers (SPs) are considered in an online social network environment whose download and upload speeds are continuously monitored for a period of specific time interval and it is observed that the average download speed offered by the SPs in the social network environment is 15.69 Mbps and the upload speed is varying from 50 percent to 70 percent of the download speed of the respective SPs.

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 complete set of data used in the proposed simulation study are given in the Table 1. 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 video sharing simulation environment is implemented using Model-View-Template (MVT) based architecture [9] where the "Models" provide an abstraction layer for structuring and manipulating data of Web application, "Views" encapsulate the logic responsible for processing a user's request and returning the response and "Templates" provide a designer friendly syntax for rendering the information to be presented to the user.

TABLE I. VIDEO SHARING TRAFFIC GENERATION – 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_v	Size of a video	2.0 to 7.0 MB
S_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 video sharing initiations	200 to 2500
D_{Speed}	Download Speed of the Network	1.5 to 30.0 Mbps
U_{Speed}	Upload Speed of the Network	50 to 70 percent of the D_{Speed}

C. Configuring Video Sharing Services in Django Web Framework

The Django "admin" is used to create a project named "VideoSharing" using the command "django-admin startproject" with an application registered as "VideoSharingApp". The python script, "settings.py" defined within the project enumerates the default backend database and all the registered applications as detailed below:

```
#settings.py
DATABASES = {
'default':{ 'ENGINE': 'django.db.backends.mysql'
'NAME': 'VIDEOSHARE_DB', 'USER': 'username'
'PASSWORD': 'password', 'HOST': '192.168.1.110',
'PORT': '3306',}}
.....
INSTALLED_APPS = ['django.contrib.admin', ….,
'VideoSharingApp',]
```

The python script, "manage.py" defined within the project starts the Webserver, migrates and synchronizes and flush the databases if required. The script, "urls.py" defines the URL patterns to link the Views. The Views render the request / response to the Templates as XML for processing and HTML for presentation. The application, "VideoSharingApp" describes and defines the required services in the Views. In the proposed cloud framework, the Model represents the Storage as a Service layer (virtual storage), the View represents the Application as a Service layer (SaaS) and the Template represents the presentation tier which includes Computational Engine.

1) Model

The logical schema for various database tables and "Filtered Data Views" used in simulation are implemented as Model entities under Django Web Framework, where each Model maps to a single database table dynamically. Model

is the single definitive source of information about the data and it is defined in the "models.py" script. The video metadata of the individual user's local store, the metadata for the registered users, the metadata used for "All Traffic Database" are described in the Model as follows:

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 Information Component)
[Sender, Video_ID, Size, Duration, Video_Frame, Receiver, Initiation_Timestamp]

push_notifications_accepted:
("acceptance database" in Information component)
[Sender, Video_ID, Receiver, Size, Duration, Video_Frame, Mem_Usage_Status, Data_Usage_Status, Relocation_Timestamp]

push_notifications_deferred:
("deferral database" in Information Component)
[Sender, Video_ID, Receiver, Size, Duration, Video_Frame, Memory_Usage_Status, Data_Usage_Status, Relocation_Timestamp]

The push_notifications_accepted and the push_notifications_deferred databases are also defined in the "models.py".

2) View

In general, the View retrieves data according to the path parameters defined in the "URL patterns" list, loads a template, renders the template with the retrieved data and returns a Http Response instance as output. Each View is a python service and an appropriate View is chosen by examining the URL that is requested as per the configurations made in the URL patterns. The codes pertaining to choose randomly a sender, video file, play duration, size, receiver, initiation timestamp etc., are defined as python functions in the views sub-directory of the application.

The video sharing traffic is generated by the end user (sender) by the way of creating a dataframe and passing it to the server application for further processing using the required parameters, which are generated randomly. For each sharing of video, the application creates a "Filtered Data View" based on the records from all traffic database filtered using the Receiver credential as in the data frame received by it.

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.

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. A separate service (thread of control) is executed by the framework for each data frame in the "Filtered Data View" after transactions have been segregated. If any one of the video frame in the "Filtered Data 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 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 python scripts for capturing the first frame and comparing the video frames are also defined in the views sub-directory as services and appropriately linked using the URL patterns.

3) Templates

A project can be configured with one or several template engines [9]. Django defines a standard API for loading and rendering templates regardless of the backend. Loading consists of finding the template for a given identifier and pre-processing it, rendering means interpolating the template with context data and returning the resulting string. The "templates" sub-directory should be created within the application directory by the end user manually. The template "dataframe.html" to receive the response from the "CreateDataFrame" service is defined as follows:

```
<html>
<body bgcolor="#bg99FF">
<p>Sender:{{sender}}</p>
<p>ReceiverList:{{ receiverList }}</p>
<p>Video_ID: {{videoID}}</p>
<p>VideoSize:{{ videoSize }}</p>
<p> PlayDuration: {{playDuration}}</p>
<p> VideoFrame: {{ videoFrame }}</p>
<p> FrameSize: {{ videoFrameSize }}</p>
<p>Ini8tiationTimestamp:{{timestamp}}>/p>
</body>
</html>
```

The corresponding URL patterns entry is mentioned in the "urls.py" as follows:

```
from django.contrib import admin
from django.urls import path
from import .views
urlpatterns = [
  path('admin/', admin.site.urls),
  path('generateTraffic/',
        views.generateVideoSharingTraffic),
  path('dataframe/',views.createDataFrame),
  path('captureFirstFrame/',
         views.captureFirstFrameFromVideo),
  path('validateFrame/',
          views.validateFrameInFilteredDataView),
  path('analysis/',views.statisticalAnalysis),
  ……]
```

To communicate between various services, the request as well as response data have been generated as XML for which the corresponding schemas are defined in the "views" subdirectory. The XMLized representation of the dataframe initiated by the sender during video content sharing is given below. In accordance with the response of the "Validation" service, the "DecisionMaking" service generates push notification acceptance or deferral data in the XML form. Similarly, when communicating between services appropriate XMLized request or response has been generated accordingly. If the attributes of any database have been changed, the corresponding Model updates the entries in the XML representation dynamically through its Views.

```
<!--Dataframe representation-->
<?xml version="1.0"?>
<Dataframe>
<Sender>User1_70-77-81-13-2A-F0</Sender>
<Video_ID>70-77-81-13-2A-F0_vid001.mp4</Video_ID>
<Receiver>User2_3C-A8-2A-AD-80-47</Receiver>
 <Video_Size>2.5</Video_Size>
 <Video_Duration>144</Video_Duration>
 <Video_Frame>70-77-81-13-2A-F0_vid001_Frame.jpg
 </Video_Frame>
 <Initiation_Timestamp>23-November-2017 12:28:41
 </Initiation_Timestamp>
 </Dataframe>
```

4) Deployment of Video Sharing Application on Google Cloud Platform

Upon testing the Video Sharing Django Application in the local machine with the local SQL server running in the backend, the application is deployed in the Google App Engine Standard Environment. For deploying the application in the Cloud, a Google Cloud Platform (GCP) project is created in the GCP console. Further, the respective operating system's compatible Google Cloud SDK Shell is installed in addition to the Python Google-Cloud client libraries in the local development environment. The APIs are enabled in the GCP console and the Cloud SQL Proxy, which provides a secure access to the Cloud SQL is also installed with respective to the operating environment. The deployment of the application in Google Cloud is explained in the sequence of steps mentioned below:

a) Creation and Initialization of Cloud SQL Instance

Google Cloud provides Cloud SQL as Storage as a Service (SaaS), which supports all the database transactions with respect the application to be deployed in the Cloud. For the creation of the Cloud SQL instance, the "tier" (called as Machine Type) and "region" are to be selected. The Machine Type determines the resources available for the Cloud SQL Instance namely memory, virtual cores etc. The "region" is the GCP region where the instance will be located. Depending on the workload of the application corresponding Machine type shall be selected. The following commands are executed in the Google Cloud SDK Shell for creating and initializing the Cloud SQL instance [11]:

- "gcloud sql tiers list" – provides the list of all the available tiers and machine types.
- "gcloud sql instances create [INSTANCE NAME] --tier = [MACHINE TYPE] --region = [REGION]" – creates the Cloud SQL instance, once created successfully from the GCP console, the details with respect to the memory allocated, connection name etc., can be verified.
- "gcloud sql instances describe [INSTANCE NAME]" – provides the detailed description of the Cloud SQL instance.
- "cloud_sql_proxy.exe –instances = ["CONNECTION NAME"] = tcp:PORT" – initializes the Cloud SQL instance and establishes a connection from the local development environment to the Cloud SQL instance.
- In the Cloud SQL instance, create a database and a user through GCP console.

b) Configuring the application with the Google Cloud

The configurations for the Database dictionary in "settings.py" with MySQL are set accordingly as per the Cloud SQL Instance connection name, database user, password and port.

c) Execution of the application in the local development environment

Before deploying the application, verification of the same is carried out in the local development environment by following the standard Web application execution procedure defined by Django Web Framework. The migrations are also carried out to set up the Models.

d) Deploying the application on the Google App Engine Standard Environment

All the static files of the application are gathered in to a single directory by executing the command "python manage.py collectstatic". These static files are moved to the production site while deploying the application to the Google Cloud. The "requirements.txt" file is created to mention the dependencies and "app.yaml" which contains the environment, runtime and entry point is also created and

finally, the application can be deployed on to the Google Cloud by executing the command "gcloud app deploy".

V. Results and Discussion

A "StatisticalAnalysis" service is defined in the Views and an appropriate path to call the service has been declared as URL pattern in "urls.py". The response of this service is rendered as XML and notified to the template for further processing. 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 analysed prior and after application of the proposed model. The parameters of the simulation model used to analyse the traffic flow across social media applications are given in Table 1.

Twelve test cases have been considered for simulation based on the attribute values U = 15, V=100, P = 30 days and the other attributes S_v, T, S_f, N, M, D_{Speed}, U_{Speed} whose values are in the range as specified in the Table 1. It is observed that the number of push notifications accepted progressively decreases with the increase in the number of video transactions which refers to the increase in the number of deferred push notifications as shown in Figure 2.

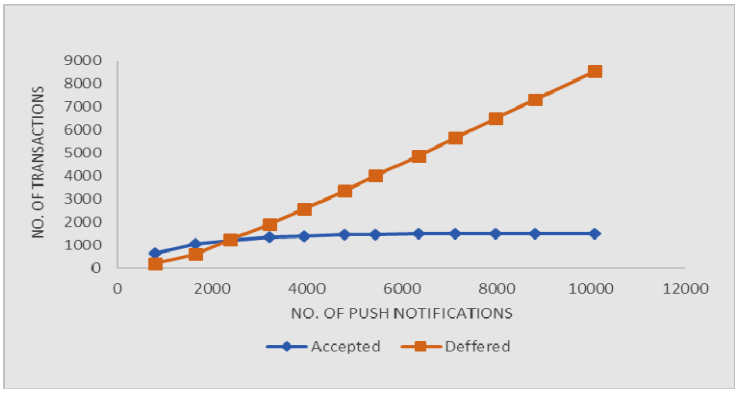


Figure 2. Total Transactions and Accepted vs Deferred Push Notifications

From the results shown in Figure 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 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 in the social media environment. 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 stable and all the incoming notifications will be deferred resulting to avoidance of identical contents. A second set of test cases are considered for simulation with the increase in the number of registered users and videos for sharing with U = 50, V = 200, P = 30 days, number of users in a group is set to a maximum of (M) 20 users, and the other attributes S_v, T, S_f, N, D_{Speed}, U_{Speed} whose values are in the range as specified in Table 1. The results obtained with respect to the data usage in the proposed model by validating the identical contents are shown in Figure 3.

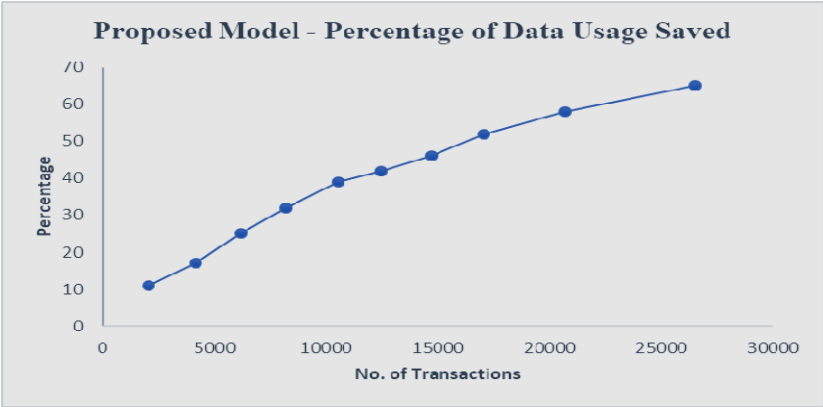
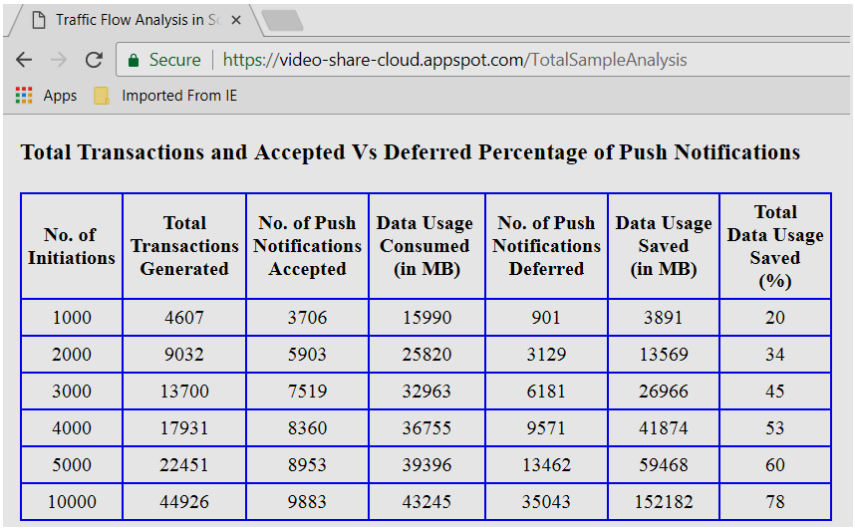


Figure 3. Saving of Mobile Data Usage by the Proposed Model.

A third set of test cases are considered for simulation with the attribute values U = 50, V = 200, P = 30 days, number of users in a group is set to a maximum (M) of 8 users, and the other attributes S_v, T, S_f, N, D_{Speed}, U_{Speed} whose values are in the range as specified in the Table 1. The analysis of total generated traffic is carried out by the Statistical Analysis service and the results are rendered to the "TrafficFlowAnalysis" template which can be accessed through the cloud environment by specifying the URL https://video-share-cloud.appspot.com/TotalSampleAnalysis and the results are displayed in the form of HTML as shown in the Figure 4.

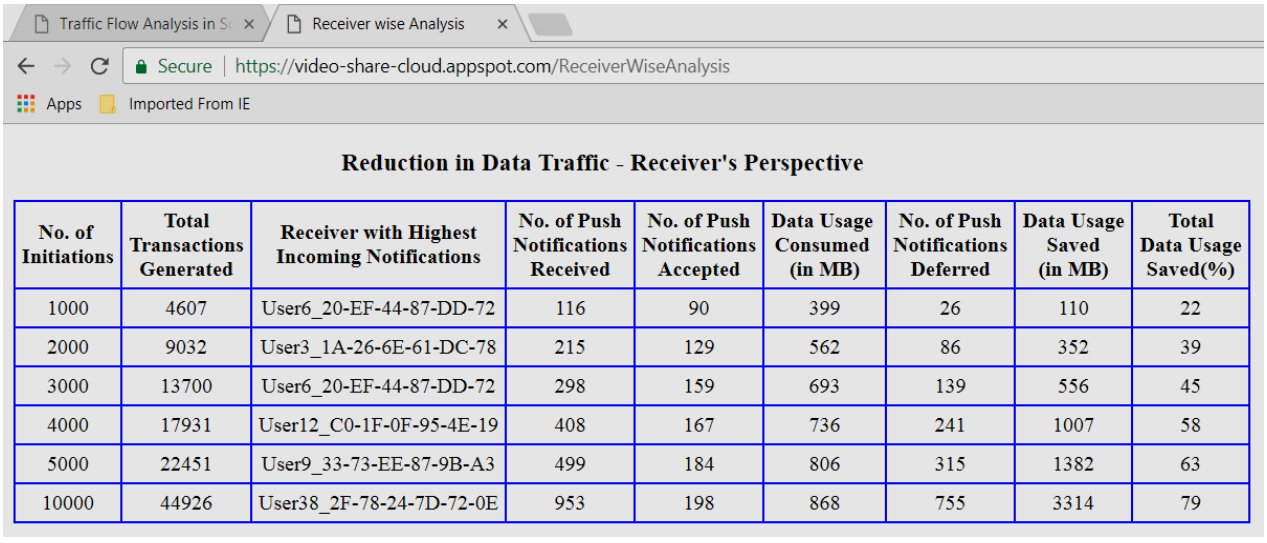


Total Transactions and Accepted Vs Deferred Percentage of Push Notifications

No. of Initiations	Total Transactions Generated	No. of Push Notifications Accepted	Data Usage Consumed (in MB)	No. of Push Notifications Deferred	Data Usage Saved (in MB)	Total Data Usage Saved (%)
1000	4607	3706	15990	901	3891	20
2000	9032	5903	25820	3129	13569	34
3000	13700	7519	32963	6181	26966	45
4000	17931	8360	36755	9571	41874	53
5000	22451	8953	39396	13462	59468	60
10000	44926	9883	43245	35043	152182	78

Figure 4. Total Transactions and Accepted vs Deferred Percentage of Push Notifications

Further, the effect on the receiver side based on the total generated traffic during simulation has also been analysed using "StatisticalAnalysis" service and the results are rendered to the "ReceiverWiseAnalysis" template which can be accessed through the cloud environment by specifying the URL video-share-cloud.appspot.com/RecieverWiseAnalysis and the results are displayed in the form of HTML as shown in Figure 5 with respect to the amount of reduction in the redundant data traffic at the network layer.



Reduction in Data Traffic - Receiver's Perspective

No. of Initiations	Total Transactions Generated	Receiver with Highest Incoming Notifications	No. of Push Notifications Received	No. of Push Notifications Accepted	Data Usage Consumed (in MB)	No. of Push Notifications Deferred	Data Usage Saved (in MB)	Total Data Usage Saved(%)
1000	4607	User6_20-EF-44-87-DD-72	116	90	399	26	110	22
2000	9032	User3_1A-26-6E-61-DC-78	215	129	562	86	352	39
3000	13700	User6_20-EF-44-87-DD-72	298	159	693	139	556	45
4000	17931	User12_C0-1F-0F-95-4E-19	408	167	736	241	1007	58
5000	22451	User9_33-73-EE-87-9B-A3	499	184	806	315	1382	63
10000	44926	User38_2F-78-24-7D-72-0E	953	198	868	755	3314	79

Figure 5. Proposed Model – Reduction in Redundant Data Traffic

The percentage saving of mobile data usage for individual receiver based on few samples of the user who received highest number of notifications in each test case is depicted in Figure 6.

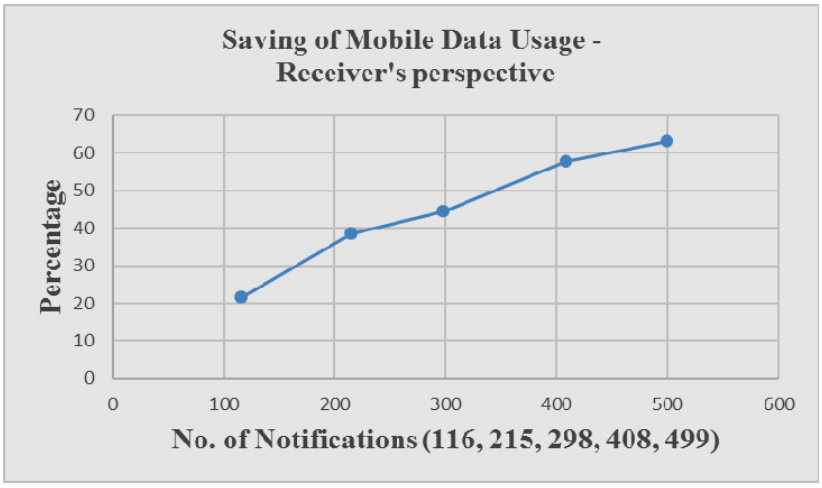


Figure 6. Proposed Model – Saving of Mobile Data Usage – Receiver's Perspective

The overhead for transmitting the first frames of the actual video contents are 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 and for this reason it is inferred the overhead for the transmission of the first frames of the corresponding 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.

Further analysis is made with respect to the network bandwidth optimization achieved from the proposed model based on the upload speed for a sender and download speed of the receiver. The maximum and minimum download speeds observed in the simulated environment is 30Mbps and 1.5Mbps respectively and the upload speed is 50 percent to 70 percent of the download speed considering the same simulated environment. The results have shown an average download speed of 15.69Mbps considering both acceptance and deferred push notifications. The amount of data usage consumed before and after validating the identical contents to restrict the redundant data traffic is shown in Figure 7.

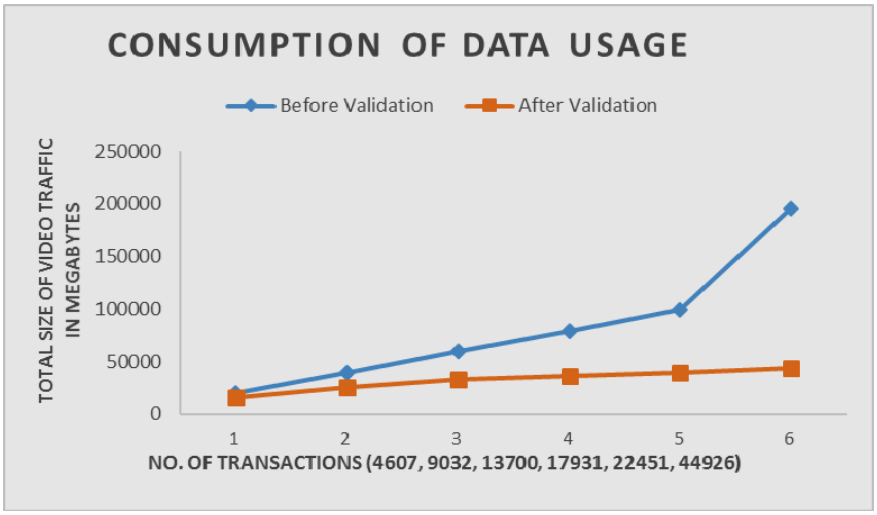


Figure 7. Proposed Model – Consumption of Data Usage Statistics.

It is inferred that, when the redundant data traffic is eliminated or reduced, the average download speed is increased to 18.61Mbps during the simulation period. The developed cloud model opens a new avenue to have a virtual content provider through which the demand of the time variant user centric contents can be analysed to predict the trending video contents for proactively caching the contents to the end users during off peak hours. Further, the analysis can be extended to understand the users' behavioural trends, for example, categories of videos shared with respect to geographical area (PLMN area) of the network, by doing this it is easy to infer the location-based trending scenario of the videos.

VI. CONCLUSION

The cloud-based simulation model is implemented using Python Django Web Framework and deployed in Google Cloud Platform. The proposed cloud-based services are highly reliable, scalable and autonomic to support ubiquitous access. The social media applications those are capable of handling multimedia contents are considered for simulation study. 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 cloud based 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 15 percent of the videos are shared multiple times across various users over a period and the percentage of deferred push notifications increases to nearly 80 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 15 to 67 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. Further, it is observed that the proposed model provides reduction of network bandwidth usage due to the optimal data traffic flow across packet data network thus increasing the download and upload speed of various Mobile Network Operators and Internet Service Providers and further 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] Chenchen Yang, Yao Yao, Zhiyong Chen, Bin Xia, "Analysis on Cache-Enabled Wireless Heterogeneous Networks", IEEE Transactions on Wireless Communications, Vol. 15, No. 1, pp. 131-145, 2016.

[2] Engin Zeydan, Ejder Bastug, Mehdi Bennis, Manhal Abdel Kader, Ilyas Alper Karatepe, Ahmet Saleh Er, and Merouane Debbah, "Big Data Caching for Networking: Moving from Cloud to Edge", IEEE Communications Magazine, Vol. 54, No. 9, pp. 36-42, September 2016. doi:10.1109/MCOM.2016.7565185.

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

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

[5] Ejder Bastug, Mehdi Bennis, and Merouane Debbah, "Living on the Edge: The Role of Proactive Caching in 5G Wireless Networks", IEEE Communications Magazine, Vol. 52, No. 8, pp. 82-89, August 2014.

[6] Xiaofie Wang, Tarik Taleb, Adlen Ksentini, Victor C.M. Leung, "Cache in the Air: Exploiting Content Caching and Delivery Techniques for 5G Systems", IEEE Communications Magazine, Vol. 52, No. 2, pp. 131-139, February 2014.

[7] Myoungjin Kim, Hanku Lee, "SMCC: Social Media Cloud Computing Model for Developing SNS Based on Social Media", Proceedings of the International Conference on Hybrid Information Technology, CCIS 206, Springer-Verlag Berlin Heidelberg, pp. 259 -266, 2011

[8] M. Balasingh Moses, V. Ramachandran and P. Lakshmi, "Cloud Services for Power System Transient Stability Analysis", European Journal of Scientific Research, Vol.56, No.3, pp. 301-310, 2011

[9] Django Software Foundation. (2018), "Django Documentation", available at: https://media.readthedocs.org/pdf/django/latest/django.pdf

[10] Google Cloud. Running Django on App Engine Standard Environment. https://cloud.google.com/python/django/appengine. (accessed on 14 March 2018).

[11] Google Analytics. https://www.monsterinsights.com/how-to-analyze-social-media-traffic-with-google-analytics/ (accessed on 14 March 2018).

[12] Neil T. Spring and David Wetherall, "A Protocol-Independent Technique for Eliminating Redundant Network Traffic", Proceedings of the ACM SIGCOMM'00, Stockholm, Sweden, pp. 87-95, 2000.

[13] Wataru Yokota, Kenji Ichijo, and Akiko Narita, "Evaluation of the Redundant Traffic Reduction Node Using the Packet Cache Coping with Different Byte Offsets among Streams", Proceedings of the 2016 International Conference on Networking and Network Applications (NaNA), Hakodate, Japan, pp. 227-232. 2016.