Face recognition in the Edge Cloud

Nasif Muslim
Dept. of Computer Science & Engineering
United International University
Dhaka, Bangladesh
nasif@cse.uiu.ac.bd

Salekul Islam
Dept. of Computer Science & Engineering
United International University
Dhaka, Bangladesh
salekul@cse.uiu.ac.bd

ABSTRACT

Cloud computing opens the door of a new paradigm where it is possible to provide service using Internet thus removing the barrier of limited hardware configuration (i.e., processing and storage) of the mobile device. *Edge cloud computing* is an emerging field within the *Cloud computing* paradigm where the server is situated at the edge of the network instead of a distant centralized data center to reduce latency. In this paper, the processing time of the face recognition system in the edge server has been measured and compared with the smartphone. The experimental results demonstrate that face recognition at the edge server performs comparably faster and it scales up better as the number of faces increases in the test image data set.

CCS Concepts

• **Computer systems organization→Cloud computing**

Keywords

Edge cloud computing; mobile cloud computing; face detection

1. INTRODUCTION

Cloud computing is a collection of infrastructure, platform and software which is used to deliver computing and storage services over the Internet to the user [5]. It allows the users to store their own information which can be accessed from anywhere in the world using any devices. Figure 1 shows the types of services provided by the Cloud. Furthermore, it allows the user to run their own application on a more powerful hardware configuration. For convenience, public Cloud computing providers set up data centers around the world. However, the centralized architecture of the Cloud computing creates some unique challenges. First problem is the propagation delay due to the geographical location of the Cloud which makes it difficult to use for real-time applications. The second problem is the transfer of application and system control from the user to the Cloud which requires complete trust between the user and the Cloud. The third problem is the lack of human-centered application deployment opportunity which could lead to the novel architecture and application.

ICISPC 2017, July 26–28, 2017, Penang, Malaysia

© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5289-5/17/07…$15.00
DOI: https://doi.org/10.1145/3132300.3132310

The Edge Cloud computing paradigm proposed to move the frontier of computing applications, data and services away from the centralized Cloud to the edge of the network. This paradigm retains the main advantage of Cloud computing. Edge Cloud consists of multiple smaller, generic Clouds situated at the Edge of the network which will be implemented in partnership with the ISPs [9]. The key advantage of the Edge Cloud is that, it improves the data transfer rate between the client and the server by reducing the latency. It also breaks the service vendor lock-in, enhances security and access to the local content, which allows the development of human-centered novel architecture and application [6]. Islam et al. [9] proposed the deployment of an Edge Cloud, integrating a variety of user-side as well as server-side technologies with Content-Centric Networking (CCN) [15] functionality. Edge-based processing allows the user to control different sources of content into customized combination (e.g., mash-ups [24]). Figure 2 shows the Edge Cloud architecture.

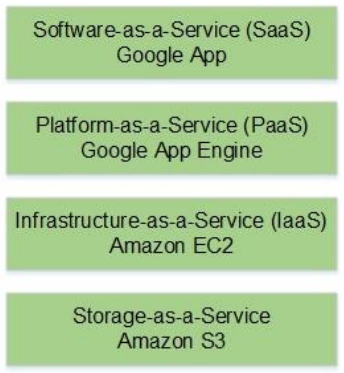
Software-as-a-Service (SaaS)
Google App
Platform-as-a-Service (PaaS)
Google App Engine
Infrastructure-as-a-Service (IaaS)
Amazon EC2
Storage-as-a-Service
Amazon S3

Figure 1. Different types of Cloud service.

Face recognition applications are used to automatically identify a face from an image. These applications are useful for personal cognitive assistance. For example, a person with cognitive and visual impairment could easily recognize a face with the aid of a face recognition application and get necessary information like address, phone number using face recognition software. It is also helpful for a normal person, who can use this information to avoid awkward social situation. Additionally, face recognition application is useful for surveillance and security. For example, at a security outpost, a visitor's face can be used to check his authentication. Running face recognition application in a mobile device (e.g., smartphone, tablet) provides better flexibility and real time results. These devices are becoming more powerful for taking photos with improved camera and sensing capabilities.

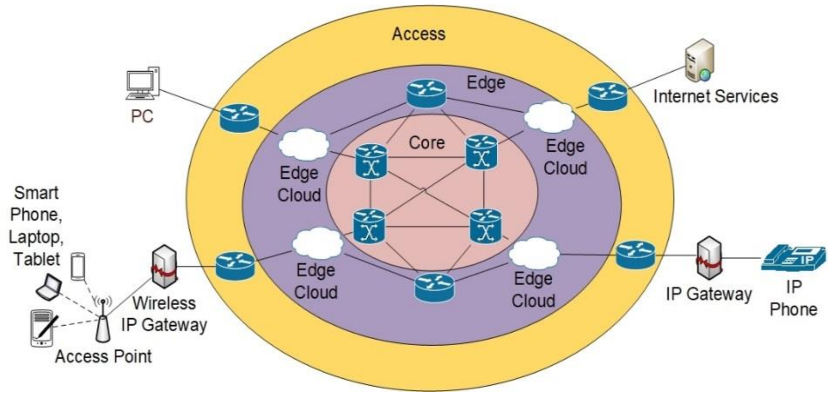

Figure 2. ***Edge Cloud*** **architecture [9].**

Face recognition application requires significant amount of processing and a large database with which the selected image taken by the smartphone will be compared. With the availability of cheap bandwidth and fast Internet speed, the face recognition computation could be transferred to the Cloud to get faster results, which has powerful processing capability and big storage facility.

This paper presents the experimental results to investigate the processing time of the face recognition system at the Edge server and the smartphone. Face recognition system is deployed both in the Edge server and the smartphone. The processing time has been measured and compared. The rest of the paper is organized as follows. At first, overview of the previous work related to Mobile Cloud computing and Edge Cloud computing are presented. Next, the face recognition system in the Edge server and the smartphone are explained. Then, experimental results are presented and evaluated. Finally, a brief summary of the results is discussed in the conclusion.

2. PREVIOUS WORK

The objective of the Mobile Cloud Computing (MCC) is to provide improved user experience to the mobile device users in terms of computation time, storage and communication. Figure 3 shows the architecture of MCC. The main challenge of MCC includes large number of mobile devices, low latency, small data bursts, and low power consumption. To overcome these challenges, Chun et al. [4] proposed Clone Cloud architecture which allows seamlessly offload execution from the mobile device to the Cloud. Hu et al. [8] proposed to deploy Cloud servers at the edge of the mobile network to reduce the latency to ensure highly efficient network operation.

Cloud computing based applications, such as P2P client [7] use a virtual client which allows the processing closer to the user for lower latency. On the other hand, Edge Cloud uses a lightweight client with minimum requirements. To support the lightweight client, a new layer *surrogate* is introduced on the top of the core Cloud service (computing, storage). From the user's point of view, the *surrogate* is the specific gateway to receive various computing and content specific services. Introduction of this layer allows to move away from enterprise-based data intensive use of the Cloud to meet the need for general public.

Live content streaming is different from traditional content delivery because live programs (e.g., live broadcasts such as sport events, concerts) cannot be prepared ahead of time [23]. The servers do the transcoding and relay the stream to the clients. To test the suitability of the Edge Cloud a prototype [10] was developed which transcodes live audio or video stream inside the Edge server. It was implemented for two different environments: in a local laboratory and in the Amazon EC2 public Cloud. The performance was measured by analyzing the inter-arrival jitters.

To liberate mobile devices from severe resource constraints, Satyanarayanan et al. [16] proposed an architecture in which a mobile user utilizes virtual machine (VM) technology to instantiate customized service software on a nearby Cloudlet. A

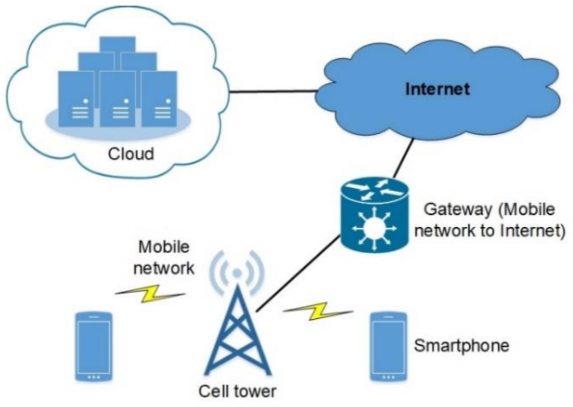

Figure 3. ***Mobile Cloud*** **computing (MCC) architecture.**

Cloudlet is a trusted, resource rich sever or cluster of servers, which is available to provide service to the mobile users via the Internet. The advantage of this architecture is that, it can provide fast request-response between mobile and Cloudlet due to the Cloudlet's one-hop physical proximity. Additionally, VM technology can provide rapid customization of infrastructure for diverse applications. Beck et al. [2] discussed several applications for the deployment at the mobile edge and classified them based on technical metrics: power consumption, delay, bandwidth usage and scalability. Kumar et al. [13] showed that all the applications are not energy efficient for MCC. For example, between two different applications: chess game and image retrieval, chess game is more beneficial for offloading to the Cloud. Namboodiri et al. [14] proposed an analytical model which can be used to generalize the energy consumption by the application in the Cloud and the mobile device. Using this model, it is possible to make a decision whether Cloud or local processing is preferable. Karim et al. [12] proposed an algorithm to off-load the computation from the mobile device to the Cloud based on network conditions, status of the battery and user inputs.

Tian et al. [18] proposed a Cloud robot system which can recognize face in real time by using the superior computational resource of the Cloudlet. The camera of the robot takes a photo and uploads it to the Cloudlet for face recognition using Wi-Fi and Bluetooth. To minimize the overall delay of the face detection and the face recognition, Soyata et al. [17] proposed mobile-Cloudlet-Cloud architecture called MOCHA where the Cloudlet receives the computation task from the mobile device and partitions the task among different Cloud service providers which are distributed over different geographical locations. The simulation result shows that the overall delay decreases as the number of Cloud server increases. Fog Computing is another paradigm which extends the Cloud computing and services to the edge of the network [3]. It is more suitable to the context of Internet of Things (IoT).

3. FACE RECOGNITION SYSTEM

Face recognition system consists of mainly two phases: face detection and face recognition. In the face detection phase, the potential location of the face is detected within an image. The face recognition phase compares the detected face with the stored face images in the database for recognition. The face recognition application is deployed in the smartphone and the Edge server to measure the processing delay of the face recognition system.

In the smartphone, first, the face recognition classifier is trained using face dataset. Then, the face recognition system takes a photo of a single person or multiple persons using its camera and starts the face detection process. The detected face is used for recognition to retrieve individual personal information like person's name. If the face is recognized then the face is marked on the photo and labeled with the person's name. Figure 4 shows the block diagram of the smartphone face recognition system.

Instead of processing the face recognition internally, the smartphone using the Internet could send the photo to the Edge server for recognition. Analogous to the smartphone, the face recognition classifier is trained in the Edge server and waiting to provide face recognition service. After receiving the photo from the smartphone, it performs the face detection and recognition steps and sends back the photo marked and labelled to the smartphone. Figure 5 shows the block diagram for the Edge server face recognition system.

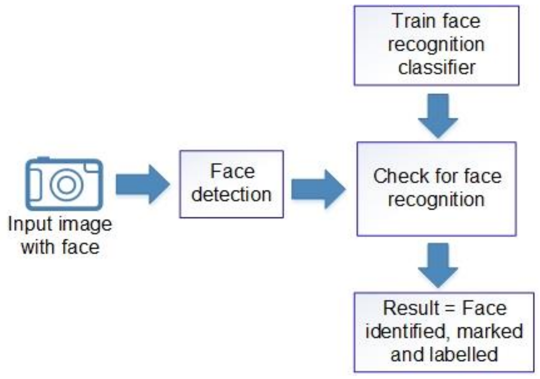
Train face recognition classifier
Input image with face
Face detection
Check for face recognition
Result = Face identified, marked and labelled

Figure 4. Smartphone face recognition system.

3.1 Face detection

Haar-cascade classifier algorithm [22] is used for face detection which has an accuracy rate of 95%. This approach starts with primitive classifiers which classify the potential face candidates into groups based on Haar features. These primitive classifiers have no computational complexity but operate on large amount of data. Using progressively more efficient classifiers based on more additional features, the algorithm eliminates some of the face candidates. The number of potential face candidates decreases at each successive stages but the complexity of the calculation increases at the same rate. As a result, the complexity of computation remains approximately same at each stage. The final stage generates the detected face with high confidence.

3.2 Face recognition

For face recognition, Local binary pattern (LBP) algorithm [1] is used. The face image is divided into small regions to extract LBP histograms and concatenated into a single feature vector of that region. These feature vectors form the efficient representation of the face which can be used to measure the similarity with the face image for recognition.

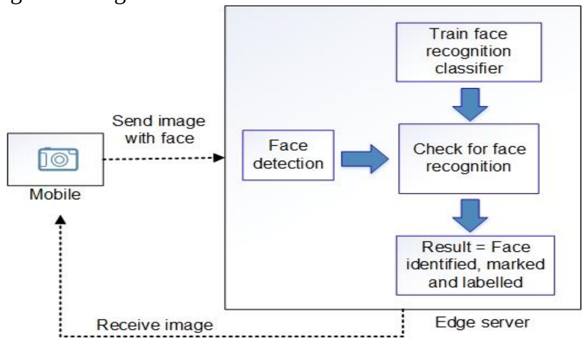
Train face recognition classifier
Send image with face
Mobile
Face detection
Check for face recognition
Result = Face identified, marked and labelled
Receive image
Edge server

Figure 5. The Edge server face recognition system.

4. PERFORMANCE EVALUATION

4.1 Experimental setup

Frontal face dataset [20] is used for training and testing the face recognition system which consists of 353 face images from 23 unique people with different lightings, expressions and backgrounds. Each image consists of 896 x 592 pixels. Android studio is used for the smartphone while Java with Eclipse IDE is used for the Edge server to develop the face recognition application.

Table 1. Platform specifications

Spec.	Edge server	Smartphone
Processor	Intel Xeon Processor E5-2630V3 8 core @2.40 GHz	Cortex-A7 Dual-core 1.2 GHz
RAM	20 GB	1 GB
OS	Windows 7	Android KitKat (4.4.2)

4.2 Experimental results

Figure 6 shows the face image set consisting of 1 to 8 faces, which is designed to compare the performance of the Edge server and the smartphone. Figure 7 shows the face detection processing time by the Edge server and the smartphone. In the smartphone the detection time goes up rapidly as the number of faces increases. On the other hand, for similar case in the Edge server, the face detection time increases slightly.

The face detection time does not depend on image file size rather it depends on the image resolution. Figure 8 shows the face detection time variation with different image file sizes (image resolution 446 x 614 pixels) for the Edge server and the smartphone. In both cases, the face detection time remains mostly invariable. Table 2 shows the face detection time for two images. Both images have 430 KB file size but different resolutions. Face detection time for the first image is higher due to the higher image resolution. Due to the image compression technique like JPEG (Joint Photographic Experts Group) [21], it is possible to reduce the image file size without compromising the image resolution. As a result, low image file size can be used for accurate face detection. Figure 9 shows the face recognition processing time in the Edge server and the smartphone. Face recognition time measurement shows similar trends. The face recognition time in the smartphone rises noticeably by the increase of the number of faces in an image. However, the face recognition time increases moderately in case of the Edge server. Figure 10 shows the comparison of total processing time between the Edge server and the smartphone, which varies over the number of faces in the image. In case of the smartphone, the total processing time is the addition of the face detection and face recognition times inside the smartphone.

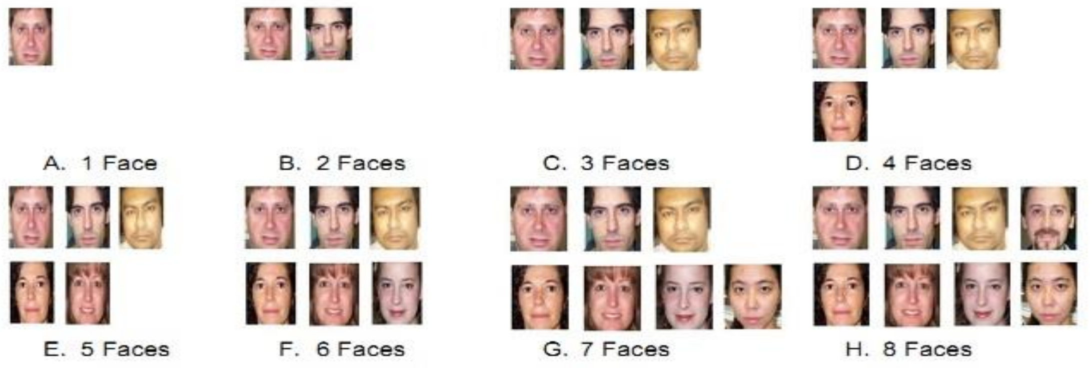

Figure 6. Face image set used for experimentation.

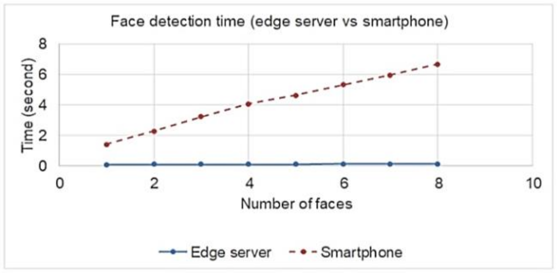

Figure 7. Face detection time in the Edge server and the smartphone.

Table 2. Face detection time varies over image resolution

	Image1 (single face)	Image2 (single face)
Image resolution	1344x1521 Pixels	896x592 Pixels
Image file size	430 KB	430 KB
Detection time	0.3227232 second	0.1520018 second

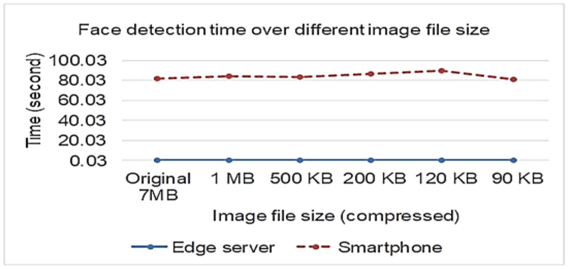

Figure 8. Face detection time variation with image file size.

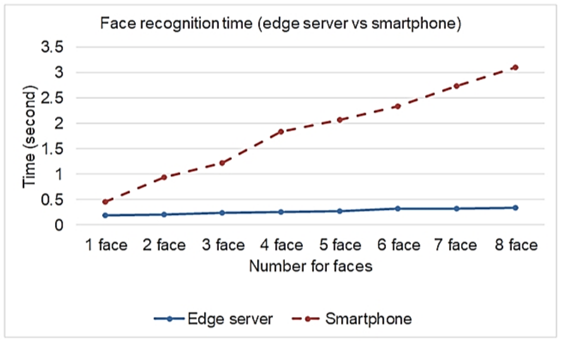

Figure 9. Face recognition time in the Edge server and the smartphone.

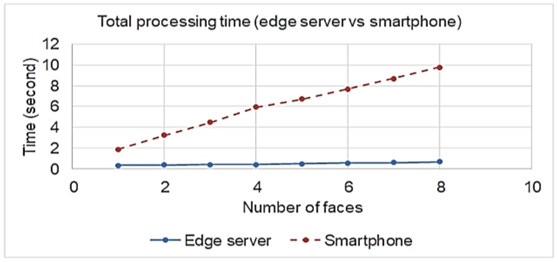

Figure 10. The total processing time in the Edge server and the smartphone

In case of the Edge server, the total processing time is the addition of the image transfer time from the smartphone to the Edge server, and the face detection and face recognition time inside the Edge server. The image transfer time depends on the available network bandwidth between the Edge server and the smartphone and the image file size. Using IPerf3 [19] tool the uploading and downloading bandwidth of the smartphone to the Edge server are measured (Table 3). The Edge server is situated 7 hops away from the smartphone as determined by the traceroute [11] tool. The mean round trip time is 10 ms.

Table 3. Bandwidth measurement using IPerf3

Path	Bandwidth (Mbps)
Smartphone to Edge server	1 .6
Edge server to smartphone	13.3

5. CONCLUSION

In this paper, the feasibility of the face recognition system in the Edge server and the smartphone has been studied. It is observed that, the face detection time depends on the number of faces in the image and the image resolution. It does not depend on the image size. Face recognition time is also influenced by the number of faces in the image since more time will be needed to recognize more faces. In both cases, the Edge server processes faster compared to the smartphone due to its better hardware specifications. Although the Edge server needs additional time for image transfer from the smartphone to the Edge server, the total processing time in the Edge server is significantly less with compared to the smartphone. Note that, this image transfer time is very low due to the location of the server at the edge of the network. In the future, based on these observations, a mathematical model will be developed which will be used to decide in runtime whether to perform face recognition computation in the Edge server or in the smartphone.

6. ACKNOWLEDGMENTS

Support for Nasif Muslim and Salekul Islam were provided in part by United International University (UIU), through its UIU Research Grant program.

7. REFERENCES

[1] Ahonen, T., Hadid, A. and Pietikäinen, M. 2004. Face recognition with local binary patterns. *Computer vision-eccv*, pp. 469-481.

[2] Beck, M. T., Werner, M., Feld, S., and Schimper, T. 2014. Mobile edge computing: A taxonomy. In *Proc. of the Sixth International Conference on Advances in Future Internet*, pp. 48-55.

[3] Bonomi, F., Milito, R., Natarajan, P. and Zhu, J., 2014. Fog computing: A platform for internet of things and analytics. In *Big Data and Internet of Things: A Roadmap for Smart Environments*, pp. 169-186. Springer International Publishing.

[4] Chun, B. G. and Maniatis, P. 2009, May. Augmented smartphone applications through clone cloud execution. In *HotOS*, Vol. 9, pp. 8-11.

[5] Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A. and Stoica, I., 2009. Above the clouds: A Berkeley view of cloud computing. Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS, 28(13), p.2009.

[6] Garcia Lopez, P., Montresor, A., Epema, D., Datta, A., Higashino, T., Iamnitchi, A., Barcellos, M., Felber, P., and Riviere, E., 2015. Edge-centric computing: Vision and challenges. *ACM SIGCOMM Computer Communication Review*, *45*(5), pp.37-42.

[7] Haque, S. A., Islam, S., Islam, M. J., and Grégoire, J. C. 2016. An architecture for client virtualization: A case study, pp. 75-89. *Computer Networks*, 100. Elsvier.

[8] Hu, Y. C., Patel, M., Sabella, D., Sprecher, N., and Young, V. 2015. Mobile edge computing—A key technology towards 5G. *ETSI White Paper*, *11*.

[9] Islam, S. and Grégoire, J. C., 2010. Network edge intelligence for the emerging next-generation internet. *Future Internet*, *2*(4), pp. 603-623.

[10] Islam, S. and Grégoire, J. C. 2012. Giving users an edge: A flexible Cloud model and its application for multimedia. *Future Generation Computer Systems*, *28*(6), pp.823-832.

[11] Jacobson, V. and Deering, S. 1989. Traceroute tool.

[12] Karim, S. A. and Prevost, J. J. 2015. August. Efficient Mobile Computation Using the Cloud. In *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on* (pp. 620-626). IEEE.

[13] Kumar, K. and Lu, Y. H. 2010. Cloud computing for mobile users: Can offloading computation save energy?. Computer, 43(4), pp.51-56.

[14] Namboodiri, V. and Ghose, T. 2012. June. To cloud or not to cloud: A mobile device perspective on energy consumption of applications. In *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, (WoWMoM)*, pp. 1-9. IEEE.

[15] Pallis, G. and Vakali, A. 2006. Insight and perspectives for content delivery networks. *Communications of the ACM*, 49(1), pp.101-106.

[16] Satyanarayanan, M., Bahl, P., Caceres, R., and Davies, N. 2009. The case for vm-based cloudlets in mobile computing. IEEE pervasive Computing, 8(4).

[17] Soyata, T., Muraleedharan, R., Funai, C., Kwon, M., and Heinzelman, W. 2012. July. Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture. In Computers and Communications (ISCC), 2012 IEEE Symposium on, pp. 59-66. IEEE.

[18] Tian, S., Saitov, D., and Lee, S. G. 2014. Cloud robot with real-time face recognition ability. *Adv. Sci. Technol. Lett*, *51*, pp.77-80.

[19] Tirumala, A., Qin, F., Dugan, J., Ferguson, J., and Gibbs, K. 2003. iperf: Testing the limits of your network.

[20] Weber, M., 2005. Computational Vision: Archive. [online] Available at: http://www.vision.caltech.edu/html-files/archive.html [Accessed 13 April 2017].

[21] Weinberger, M. J., Seroussi, G., and Sapiro, G. 2000. The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS. *IEEE Transactions on Image processing*, *9*(8), pp.1309-1324.

[22] Wilson, P. I. and Fernandez, J. 2006. Facial feature detection using Haar classifiers. *Journal of Computing Sciences in Colleges*, *21*(4), pp.127-133.

[23] Wu, Y., Wu, C., Li, B., Qiu, X., and Lau, F. C. 2011. June. Cloudmedia: When cloud on demand meets video on demand. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on* pp. 268-277. IEEE.

[24] Yu, J., Benatallah, B., Casati, F., and Daniel, F. 2008. Understanding mashup development. *IEEE Internet computing*, *12*(5).