Eyes in the Sky: A Free of Charge Infrastructure-less Mobile Ad-hoc Cloud

Mohamed Sherif
Faculty of Engineering Alexandria University
Egypt
msherif449@gmail.com

Mohamed Azab
The City of Scientific Research and Technological Applications
Egypt
mazab@vt.edu

Yamen Emad
Faculty of Engineering Alexandria University
Egypt
yamenemad4@gmail.com

Sherine Sameh
Faculty of Engineering Alexandria University
Egypt
sherine.aziz.std@alexu.edu.eg

Farida Menisy
Faculty of Engineering Alexandria University
Egypt
farida.menisy@gmail.com

Boshra Kandil
Faculty of Engineering Alexandria University
Egypt
boshrakandil94@gmail.com

ABSTRACT

The evolution of Ad-hoc Clouds has become completely inspiring. Therefore, the usage of free resources distributed around us such as 'Cloud as a Service' is witnessing a great importance. Ubiquitous computing turned out to be a fact that has many unutilized resources. These unutilized resources can be used to create Ad-hoc Cloud. In this paper, we create a novel mobile ad-hoc cloud based on a network of inter-connected low cost, small size, resource constrained and widely available devices. We present a novel application to detect criminals using low cost equipment. This way the application can be considered a method that compensates the total cost that was needed for creating the cloud infrastructure and operational cost. Simulation studies showed that the performance of the created ad-hoc cloud is acceptable in comparison with a small scale conventional cloud hosting container based distributed application.

CCS Concepts

• **Networks** → *Cloud computing*; Mobile networks; Mobile ad hoc networks • **Theory of computation** → Support vector machine

Keywords

• Cloud Computing, Ad-hoc Cloud, Linux Container, Machine Learning, OpenCV

UCC'17 Companion, December 5–8, 2017, Austin, TX, USA © 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5195-9/17/12…$15.00
https://doi.org/10.1145/3147234.3148122

1 INTRODUCTION

Using cloud computing (CC) and cloud services has been highly adopted in the past few years. What makes CC the most preferable paradigm is providing access to distributed storage, using fast inexpensive management controllers and virtualization technology. The success that cloud computing has recently attained was completely inspiring. Researchers developed an approach for gaining the most benefit of excess computing and storage resources in scattered hardware equipment [1]. For this reason, they introduced the concept of mobile ad-hoc computing referring to vehicular cloud as an application of that concept. Researchers discussed the increase in number of vehicles that roads are witnessing nowadays. Additionally, cars are supplied with powerful on-board computers and highly accurate sensors which makes applying their concept highly possible. Researchers aimed to make use of the wasted storage resources of on-board computers [2].

A mobile ad-hoc network (MANET) is a continuously self-configuring, infrastructure-less network of mobile devices connected wirelessly [3].

Each device in a MANET is free to move independently in any direction, and will therefore change its links to other devices frequently. Each must forward traffic unrelated to its own use, and therefore be a router. The primary challenge in building a MANET is equipping each device to continuously maintain the information required to properly route traffic [4]. Such networks may operate by themselves or may be connected to the larger Internet. They may contain one or multiple and different transceivers between nodes. This results in a highly dynamic, autonomous topology.

In this paper, we create a novel mobile ad-hoc cloud based on a network of inter-connected low cost, small size, resource constrained and widely available devices. The decision of renting-out unutilized resources is economically appealing to the authors. Accordingly, we present a novel idea of a two-in-one system mixing between two smart applications operating on a conventional, extremely low-cost mobile IoT board. As an example, many criminals manage to escape since police resources are not sufficient for pervasive and fast search. As a result, developing computer applications to detect criminals and law violations is essential. Governments invest thousands of dollars in such technologies. Accordingly, we present a complete developed application that depends on low cost based resources to help governments detect criminals. The developed system uses this application to compensate the cost of creating the infrastructure needed to create the ad-hoc cloud.

In our scenario, we assume that the entire infrastructure will be provided by third party for free to the primary user in return for the administration infrastructure support services that will be handled by a secondary user. The actual profit will be achieved by renting the unutilized resources of the platform by the presented cloud platform.

Therefore, the paper introduces a comprehensive solution with all necessary modules to serve the primary application objectives "Autonomous Detection for Criminals" including algorithms for image processing and face detection. The paper also presents a complete cloud management platform including remote deployment mechanism, suitable virtualization technique and all administration and management tools.

Our contributions in this paper can be summarized as follow:

Creating an ad-hoc cloud using unutilized resources of resource constrained IoT device with no external investments. Moreover, hosting a resource efficient, comprehensive, low cost based application with a remote cloud backbone to search for criminals in real-time making substantial profits from its operation.

The rest of the paper is organized as follow: Section 2, we give an overview of 'Eyes in the Sky'. We detail the architecture of the proposed approach, introduce various available connections and highlight the one we used to give reasons.

Section 3, we explain how face detection algorithm works in details. In Section 4, we present our evaluation. Finally, we give a comprehensive conclusion about the paper in Section 5.

2 Eyes in the Sky (EitS)

A lot of problems face our communities both locally and globally. One major problem is the increase in number of thieves and criminals in our countries. We present 'Eyes in the Sky' to help in solving this problem depending on technology based resources. These resources do not require an enormous amount of money for investment since that will not match the economic status of the Middle East.

Another application that 'EitS' presents is creating a Cloud-like Platform that is cost efficient depending on a few number of resources. It can also be a source of income or profit when hired to external users who are in a massive need for such an application due to the absence of infrastructure in the tested region. Accordingly, we present a two-in-one application.

The name 'Eyes in the Sky' is a mix of the two created applications. 'Eyes' represents the camera connected to the resource constrained device that is placed on a high-reaching object such that detecting people and their face recognition is possible. The application serves the government since it provides an affordable tool that helps in detecting criminals and their location as fast as possible. The government user is notified by the criminal's location. 'The Sky' represents the mobile ad-hoc semi-infrastructure cloud that we created using unutilized resources of resource constrained device. The developed cloud service can be used in several applications that will be discussed in later chapters. 'Eyes in the Sky' can be used as an access point due to the presence of the Raspberry Pi as a part of it and that provides Wi-Fi for nearby people.

2.1 EitS Architecture

Some resources were used for developing the criminal detection application, the remaining free storage will be used for creating a cloud using the same group of Raspberry Pi(S). In such case, we can create a cloud with a small amount of resources and a low cost instead of forcing people to use a cloud on a data center and paying a much bigger amount of money in return.

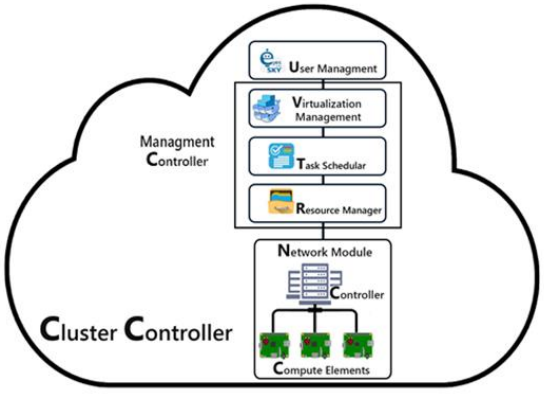

Figure 1 EITS Architecture Overview.

In this section, we offer detailed overview of the 'EitS' basic architecture and how it must be engineered. Referring to Figure 1, EITS consists of:

1. User management, which is the portal where users create new accounts to have access to our service and upload their containers.
2. Virtualization management, the main target was to ensure having an isolated operating system. User's application needs to be isolated, shared, encapsulated, easy deployed and easy revoked. Therefore, using containers was the most suitable method.
3. Task scheduler, it allocates computational tasks to individual clusters based on the amount of available resources.
4. Resource manager, it handles Raspberry Pi's allocation, resource utilization and status of each resource.
5. Network module, it can be described as follows:
   It is a group of interconnections that can be infrastructure based to maintain a connection or infrastructure-less that can work anywhere under any condition. It is composed of a controller that controls a number of compute elements. We use sockets technology to enable the connection between the nodes and the management controller.

EitS operates as follows. The management controller receives a request from the user through the developed dashboard. The task scheduler decides which raspberry pi will be chosen to execute the request depending on an algorithm developed precisely for this purpose. The algorithm checks which device is the most efficient to meet the anticipated demands of the requested services. Then, the user management system returns the results to the user.

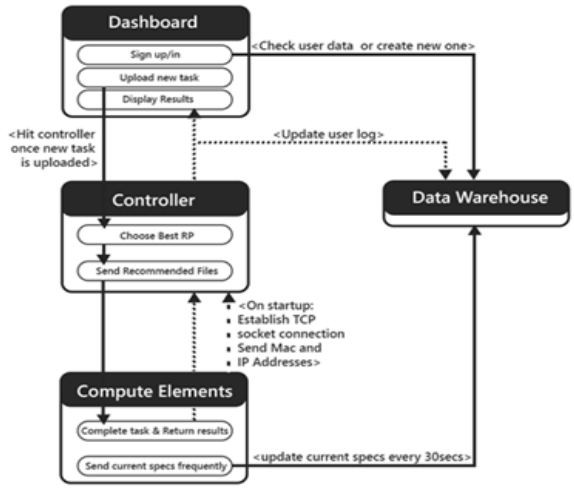

Figure 2 Network Connections.

Figure 2 describes the main modules for 'EitS' reflected in the management platform and its dashboard. The user will use the dashboard to upload new tasks. The dashboard will notify the controller once a new task is uploaded. The controller chooses the best Raspberry Pi based on the Best Pi Algorithm that will be described in section 2.2 and sends the recommended files. When the compute elements (Raspberry Pi(s)) are connected to the controller, they establish a TCP socket connection and send their Mac and IP addresses. The compute elements do the tasks and return the result to the controller which is then displayed on the dashboard. The dashboard is connected to the Data warehouse such that whenever a task is done, it is updated in the Data warehouse and the compute element that carried the task is updated to be available. The raspberry pi compute elements send their current specifications and update them in the Data warehouse every 30 seconds.

2.2 Best Raspberry Pi's mechanism

When a user makes a request, we need to find the most suitable Raspberry Pi to handle the request. Accordingly, we developed the Best Raspberry Pi Algorithm that works as follows. There is a timer event script that fetches seven criteria of the Raspberry Pi. Each one is represented as a set of points which are then added and compared with total points of other Raspberry Pi(s) available to choose the most efficient one. The Raspberry Pi that has the greatest number of points is the one chosen for carrying a certain task. These criteria shown in Figure 3 represent the Raspberry Pi's specifications which are updated periodically in the database.

We choose the Best Raspberry Pi by checking the characteristics in the following order. First, Raspberry Pi(s) with cameras have much more processing to do compared to the ones without cameras. Therefore, we

give priority to the Raspberry Pi(s) with no cameras to run the coming requests. Second, the maximum number of tasks allowed to run on each Raspberry Pi is set to three tasks. We put the coming task on the Raspberry Pi with the least number of tasks running on it. Then, we choose the Raspberry Pi that has the least CPU usage and most suitable memory available for carrying the task. Checking the temperature which has a threshold (80 F) comes next. Finally, we choose the Raspberry Pi with the least working time duration in order not to make overload on the Raspberry Pi(s).

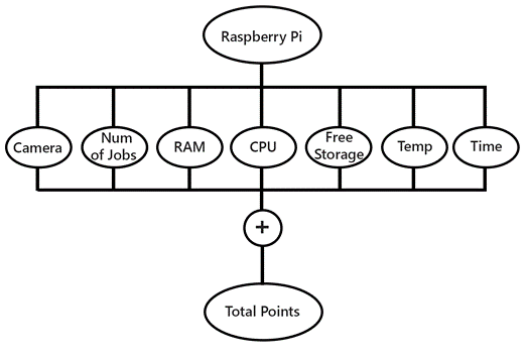

Figure 3 Best Raspberry Pi Calculation Method

2.3 Remote Management Platform

The system consists of three separate dashboards with no one having access on the other for security and privacy issues. One for the administrator, another one for the government and the last one for the normal user.

2.3.1 Administration Management

In this dashboard, only one administrator, database administrator, has access on all other admins accounts and government accounts. He has the authority to edit, add and delete other accounts meaning that if someone wants to create a government account, the database administrator must approve first, the account will not be created by just clicking register. Same applies when creating an admin account, the database administrator must approve. If a user forgot his password, he is the only one that can give the approval for creating a new password. The database administrator's dashboard shows all the available computing nodes with their full specifications and the running processes. It also shows the history of all the operations that the admins did.

2.3.2 Task Containerization

The user uploads the dockerfile containing the specifications of his application and if the user want to add files to his application he can also upload them alongside the dockerfile also the process name is name. The dockerfile must be written in this format "dockerfile" and we check for this syntax, if the file is correct then the task scheduler sends a request to the management controller which takes the dockerfile and check the right raspberry pi to send it to. The raspberry pi is configured to automatically react to the incoming docker files using the customized docker engine. It starts to build the image from the dockerfile and then run the image in an isolated container. The Raspberry Pi updates the database with the new process specs the image id, container id, port, and public IP of the container.

The management controller then sends the results back to the portal in order to the user to trace his process or to take the IP of the container to access it remotely if he wants to.

2.3.3 Criminal Detection

In this dashboard, the government user can search for criminals added previously in the database. The history of the government dashboard is open for any government user which means that he can see any change for example a government user added someone as a criminal, editing someone's crime or searching for some person as a criminal. The government user can also upload a set of images forming a training set to add a new criminal. He can also compare a picture of a person with training sets in our database to know if that person is a criminal or not. When a camera detects a criminal in a certain location, a notification will be sent to the government users stating the location of the criminal.

2.4 Networking Module

2.4.1 Infrastructure Approach

When the Raspberry Pi receives power, the 3G module is automatically defined then the client's script runs on the Raspberry Pi, so the IP address and the matching MAC address return back and added in Python Dictionary. The connection sequence can be described as shown in Figure 4. If an interruption occurred in the socket connection and the client reconnects back to the management controller, it will be considered as a new client but its static MAC address will be matched with its IP address from the Python Dictionary neglecting any change in the IP address if occurred since the Raspberry Pi is chosen according to the MAC address.

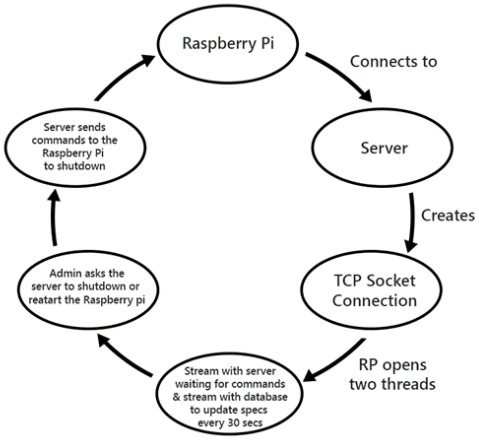

Figure 4 The Connection Sequence.

Internet Service Provider in the tested region doesn't provide real IP's so the management controller cannot know the computing elements' IP as it varies with the network. As a solution, we made a reverse direct connection where the computing element notifies the management controller in the beginning of the connection since the management controller has a static IP.

3 Criminal Detection Platform (The Secondary Application)

Due to the rapid increase of crimes. Timely detection government. (IoT) and increasingly cheap webcams can be used to facilitate this task. Generally, the application of IoT technologies to the fields of smart cities can be seen more often. In this paper, we present the design of IoT based smart crime detection system. The proposed system is able to detect crimes in real-time.

IoT technologies can be applied to smart cities in order to improve the daily life of local resident. As cameras are becoming ubiquitous in the (IoT) and can use face recognition technology which is the primitive use of our application. It adds context to our system as a low cost face recognition security system deployed on our computing nodes, aimed in detecting and tracking criminals [6].

Face recognition has been an active research topic for years. It is a series of several related challenges. Given an input image with multiple faces, face recognition systems typically first run face detection to isolate the faces. Each face is preprocessed and then a low-dimensional measurement (or embedding) is obtained. A low-dimensional representation is important for efficient classification [6]. Challenges in face recognition arise because Human's face is a dynamic object with high degree of variability in its appearance. This makes face recognition a difficult problem in computer vision and thus many challenges exist when it comes to face recognition. Humans can alter their facial features through growing a mustache or maybe wearing a veil and this might decrease the accuracy of the recognition process.

However, Today's top-performing face recognition techniques are based on convolutional neural networks such as Google's FaceNet systems that yield the highest accuracy. FaceNet directly learns a mapping from face images to a compact Euclidean space where distances directly correspond to a measure of face similarity [6]. Once this space has been produced, tasks such as face recognition, verification and clustering can be easily implemented. OpenFace was created which is an implementation of FaceNet system that is a general-purpose library for face recognition. Our experiments show that OpenFace offers higher accuracy than prior open source projects and is well-suited for our scenario.

As a human, our brain is wired to recognize everything automatically and instantly. Computers are not capable of this kind of high-level generalization, so we have to teach them how to do each step in this process separately. We need to build a pipeline where we solve each step of face recognition separately and pass the result of the current step to the next step. In other words, we will chain together several machine learning algorithms.

The first step in our pipeline is face detection using Haar Feature-based Cascade Classifiers [7]. It detects facial features and ignores anything else. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images. When it comes to face detection the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. Instead of applying all features the on a window, group the features into different stages of classifiers and apply one-by-one. If a window fails the first stage, discard it. We don't consider remaining features on it. If it passes, apply the second stage of features and continue the process. The window which passes all stages is a face region as shown in Figure 5.

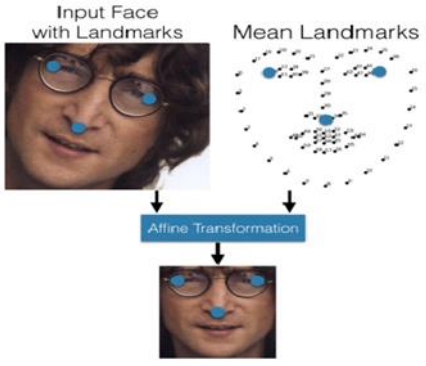

Figure 5 Preprocessing (Alignment with an Affine Transformation). The transformation is based on the large blue landmark

After, isolating the faces in the video. The face detection returns a list of bounding boxes around the faces in an image. We have to deal with the problem that faces turned in different directions look totally different to computer. As Pose and illumination have been a long standing problem in face recognition [7, 9]. A potential issue occurs when using the bounding boxes directly as an input into the neural network. To account for this, we will try to warp each picture so that the eyes and lips always appear in the same location in the image. This will make it a lot easier for us to compare faces in the next steps. To do this, we are going to use an algorithm called face landmark estimation [7].

The basic idea is to come up with 68 specific points (called landmarks) that exist on every face –the top of chin, the outside edge of each eye, the inner edge of each eyebrow, etc. Theses 68 landmarks are detected with dlib's face landmark detector. It is a machine learning algorithm which is able to find the specific 68 points on any face. Given an input face, our affine transformation makes the eye corners and nose close to the mean locations. The affine transformation perform basic image transformations like rotation and scaling. It will resize and crop the image to the edges of the landmarks so the input image to the neural network is 96 × 96 pixels as shown in Figure 6 [8].

Now no matter how the face is turned, we are able to center the eyes and mouth are in roughly the same position in the image. This will make our next step a lot more accurate.

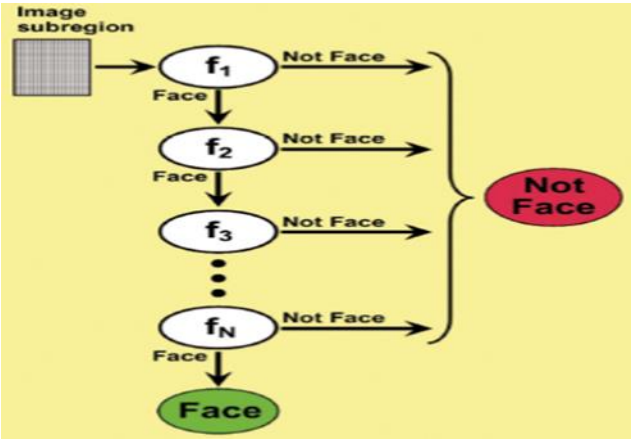

Figure 6 Illustratration of Haar cascade feature extracting.

The third step in the pipeline, passing the centered face image through a neural network that would extract a few basic measurements from each face. So, we could measure the face to be recognized (unknown face) the same way and find the known face with the closest measurements. These measurements could include either size of ear, spacing between eyes, etc. but, it turns out that measurements that Seem obvious humans don't really make sense to computers. Researchers discovered that it's better to let the computer figure out which measurements to collect by itself. As Deep learning does a better job than humans out which parts of a face are important to measure. The solution is to train a Deep Convolutional Neural Network. We are going to train it to generate 128 measurements for each face. To train, we use triplets of roughly aligned matching / non-matching face patches [6].

The training process using triplet matching algorithm can be described in the following steps. first, Load a training face image of a known person. then Load another picture of the same known person. Finally load a picture of a totally different person, this process of training a convolutional neural network to output face embedding requires a lot of data and computer power. But once the network has been trained, it can generate measurements for any face, even ones it has never seen before. OpenFace already did this and they published several trained networks which we can directly use. So all we need to do ourselves is run our face images through their pre-trained network to get the 128 measurements for each face.

It is hard to detect what part of the face are these 128 measurements. It doesn't really matter to us. All that we care is that the network generates nearly the same numbers when looking at two different pictures of the same person. OpenFace's execution time is well-suited for our scenario compared to other techniques.

Some constraints need to be introduced to our model:

- The number of images in the dataset shouldn't be less than 24 image/person.
- The criminal is labelled only by his ID.
- The system must be trained and operated using ARM architecture.
- Multiple criminals can be detected.
- Data of the criminals is sent in the form of compressed file.

The last step in the whole procedure is to find the person with the closest measurements to the test image. Any basic machine learning classification algorithm can do so. We will be using simple linear SVM classifier [7]. All we need to do is train a classifier that can take in the 128 measurements resulting from a new test image and tells which known person is the closest match. The result of the classifier is the id of the person. Upon recognizing the right face the system sends an alert to the government portal running on our management controller. The proposed system serves as a useful tool for both police agencies to determine crime and for citizens to be safe.

4 Evaluation

CloudSim is a simulation tool that allows cloud developers to test the performance of their provisioning policies in a repeatable and controllable environment, free of cost. It is a simulator; hence, it doesn't run any actual

software. It can be defined as 'running a model of an environment in a model of hardware', where technology-specific details are abstracted. CloudSim is a library for the simulation of cloud scenarios. It provides essential classes for describing data centers, computational resources, virtual machines, applications, users, and policies for the management of various parts of the system such as scheduling and provisioning.

We used CloudSim as a testbed in our experiment to evaluate some metrics such as performance, memory consumption, the effect of increasing the number of tasks on task completion time and the power consumption. CloudSim enables the modeling of datacenter, hosts, VMs, cloudlets and brokers.

Datacenter is composed of same or varied configuration hosts. The host in a datacenter is characterized by host id, RAM, storage, bandwidth, processing power (MIPS) and number of processing elements (PE). The hosts are responsible for managing the VM creation, VM migration, VM destruction and VM provisioning. The allocation of VMs to host depends on the allocation policy adopted by VM provisioner, default policy being first come first serve. The tasks (cloudlets) are handled by VMs that are allocated a share of processing power and memory on datacenter hosts. The number of VMs created on a host depends upon the VM RAM. The VMs are characterized by image size, RAM, processing power, bandwidth and number of PEs while the cloudlets are characterized by length, file input size and file output size. For task or cloudlet mapping to VM, the host utilizes the time-shared or space shared allocation policy [9].

4.1 Experiment Description

We configured a virtual machine with the specifications shown in **Table 1** with task specifications as shown in **Table 2**. These tasks are deployed on a Raspberry Pi with the specifications shown in **Table 3**.

Table 1 Virtual Machine Specifications

Specifications	Value
Image Size	1 MB
RAM	1000 MB
MIPS	1000 MB

Table 2 Task Specifications

Specifications	Value
File Size	3000 KB
Output Size	3000 KB

Table 3 Raspberry Pi Specifications

Specifications	Value
RAM	1 GB
Storage	1024*8 MB

4.2 Results

Figure 7 shows the effect of increasing the total number of hosts on the task completion time. The number of tasks is constant and the number of virtual machines "VMs" varies according to the number of hosts. The experiment is tested by changing the number of hosts from 50 to 500 therefore changing the number of VMs since each host has only one VM. The overall response time to execute the tasks is used as a metric to evaluate the performance of the cloud. As the number of hosts increases, the overall time decreases so consequently, the performance becomes better.

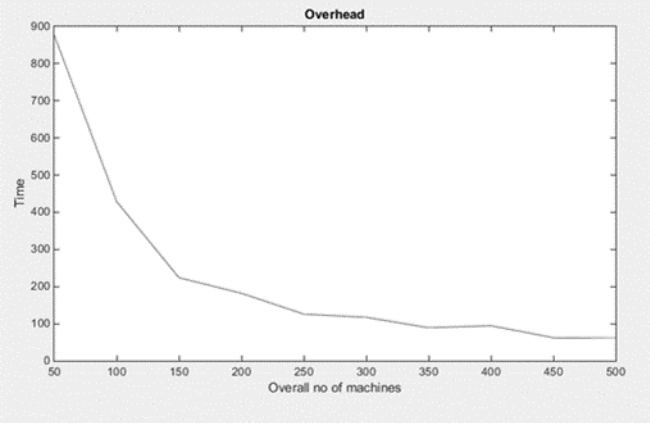

Figure 7 Number of hosts verses Number of tasks completed

Figure 8 shows the effect of increasing the total number of cloudlets (tasks) on the task completion time in time-shared policy. The number of hosts and VMs remaining constant (For instance = 100). In time-shared allocation policy, each VM receives a time slice on each processing core and similarly the tasks dynamically context switches during the VM lifetime. Multiple VM can simultaneously multi-task within a host and similarly multiple tasks can simultaneously multi-task within a VM. The experiment is tested by changing the number of tasks from 50 to 500. As the number of tasks increases, the workload increases and as a result the overall time increases.

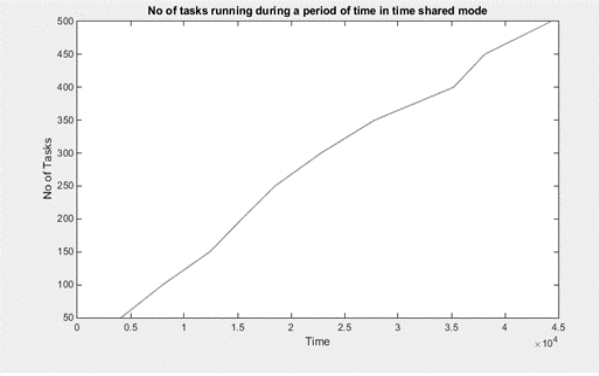

Figure 8 Total number of cloudlets Verses Task completed under time-shared policy.

Figure 9 shows the effect of increasing the total number of tasks on the memory consumption. The number of hosts and VMs remain constant (For instance = 100). The experiment is tested by changing the number of tasks from 50 to 500. We observe a linear growth with an increase in the numbers of tasks. Therefore, the memory consumption also increases.

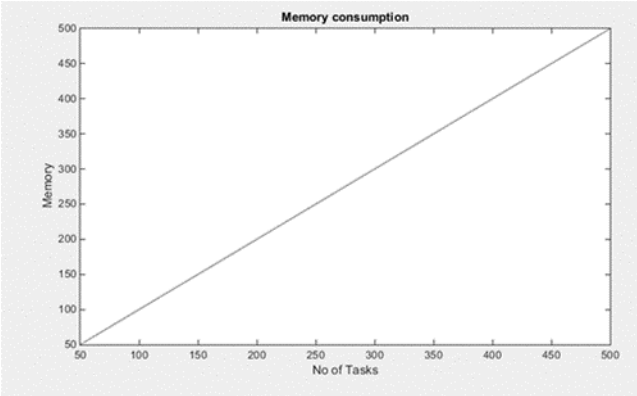

Figure 9 shows the effect of increasing the total number of tasks on the memory consumption.

Figure 10 shows the effect of increasing the number of tasks on the power consumption. The fraction of power consumed by idle management controller (k) equals 1.4 and the maximum power consumed when the management controller is fully utilized equals 3.7. The experiment is tested by changing the number of tasks from 50 to 500. As the number of tasks increases, the power increases.

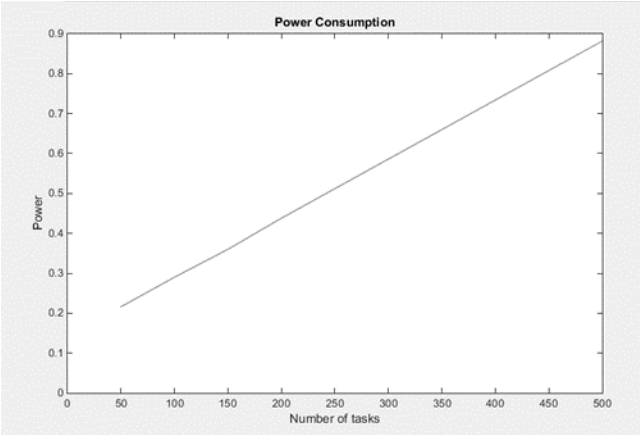

Figure 10 shows the effect of increasing the number of tasks on the power consumption.

Figure11 compares the evaluation of how the processing elements run on containers with real cloud simulation that has one virtual machine running on one host with high specifications, we concluded the following:
Both experiments almost gave the same results which indicates that the project is running successfully with satisfying performance and efficiency.

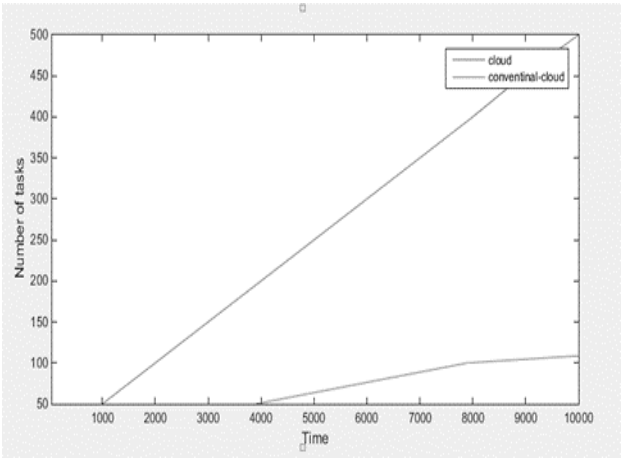

Figure 11 CloudSim versus Conventional Cloud

5 Conclusion

In this paper, we managed to successfully create ad-hoc cloud from unutilized resources and hide the cost of it by developing a secondary application. We introduced a comprehensive, free of charge, resource efficient criminal detection application that runs on a resource constrained IoT device. In the future, we want to improve the criminal detection application in a way such that a criminal can be detected among pedestrians depending on suspicious behavior. We are also looking forward to make our cameras have the ability to detect stolen cars from the processing of cars plate number.

6 References

[1] Khalifa, A., Azab, M., & Eltoweissy, M. (2014, October). Resilient hybrid Mobile Ad-hoc Cloud over collaborating heterogeneous nodes. In Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2014 International Conference on (pp. 134-143). IEEE.

[2] Arif, S., Olariu, S., Wang, J., Yan, G., Yang, W., & Khalil, I. (2012). Datacenter at the airport: Reasoning about time-dependent parking lot occupancy. IEEE Transactions on Parallel and Distributed Systems, 23(11), 2067-2080.

[3] Toh, C. K. (2001). Ad hoc mobile wireless networks: protocols and systems. Pearson Education

[3] Zanjireh, M. M., Shahrabi, A., & Larijani, H. (2013, March). Anch: A new clustering algorithm for wireless sensor networks. In Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on(pp. 450-455). IEEE.

[4] Byun, J. Y., Nasridinov, A., & Park, Y. H. (2014). Internet of things for smart crime detection. Contemporary Engineering Sciences, 7(15), 749-754.

[6] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 815-823).

[6] Kazemi, V., & Sullivan, J. (2014). One millisecond face alignment with an ensemble of regression trees. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1867-1874).

[7] Amos, B., Ludwiczuk, B., & Satyanarayanan, M. (2016). Openface: A general-purpose face recognition library with mobile applications. CMU School of Computer Science.

[8] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., & Buyya, R. (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and experience, 41(1), 23-50.