MoCCA: A Mobile Cellular Cloud Architecture

Amitabh Mishra and Gerald Masson

Johns Hopkins University Information Security Institute, Baltimore, MD, USA; e-mail: amitabh@cs.jhu.edu, masson@jhu.edu

Received 31 May 2013; Accepted 15 July 2013

Abstract

This paper presents MoCCA – a cellular cloud architecture for building mobile clouds using small-footprint micro-servers running on cell phones. We provide details of this architecture which is based on GSM standard, discuss several challenges, and include performance results to validate the assumptions that a mobile cellular cloud can indeed be in the realm of possibilities.

Keywords: Mobile cellular networks, cloud computing, performance.

1 Introduction

Cloud computing is a new computing paradigm, involving data and/or computation outsourcing, with infinite and elastic resource scalability. The NIST defines cloud computing as: "a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [15].

Cloud computing service providers typically possess large data centers consisting of a large number of servers. The resources of these servers are provisioned to the clients on demand. The cloud service providers typically provide one or more of the following services: Software-as-a-service (SaaS), Platform-as-a-service (PaaS), or Infrastructure-as-a-service (IaaS). In SaaS, the cloud provider runs applications on the cloud platform, and clients usually access these applications via a web-browser interface. Examples of SaaS are Google Docs, Microsoft Live, etc. In PaaS, clients get more access to the cloud, by deploying or configuring applications or code that runs on the cloud providers software and operating system platform. In IaaS, clients get the greatest control over the cloud. In this model, the clients can deploy operating system on virtual machines running on the cloud servers, provision resources, and run arbitrary software. Data Flow programming has gained popularity for data processing in Clouds. A widely used programming framework in this model is Google's MapReduce [5]. Given the operational advantage of using a cloud for computing, they have become widely popular in the developed world. However, if we want to bring cloud computing across the digital divide, to developing nations, we face a big obstacle. As of now, setting up a moderate sized cloud requires significant investment in the data center infrastructure. A high-density data center with 10000 servers can cost up to $4 million or more [20]. Larger scale facilities such as Microsoft Azure's Chicago data center cost $500 million [12, 16]. A large part of this cost goes towards setting up the physical infrastructure for cooling, racks etc., making it prohibitively costly for a new player to enter the cloud computing business.

To solve this problem, we propose using the widely deployed mobile phones (MS) as the building blocks for clouds. We define a mobile cloud as a cloud composed of mobile phones and the associated networking components. In contrast with traditional clouds, a mobile cloud is formed using loosely connected servers. A mobile cloud provides a lot of business advantages over traditional clouds. It can allow base station owners or cell phone operators to enter the market with little establishment cost. Base stations which are integral part of the mobile cloud architecture are already deployed in most places. To build a mobile cloud, a provider only needs to deploy additional software and minimal amount of additional hardware to the base stations. Running a mobile cloud microserver on a cell phone uses up energy and phone resources, so the clients must have some incentives to participate in such a cloud.

Unlike a traditional data center, the mobile cloud does not require extensive cooling systems, a large building for servers, wiring, and that it being close to power utilities. A mobile cloud is also self-balancing when used to serve local computing needs. For example, in a building with few cell phone users, the resources required to run a local service is low. So, a base station can run local services on the few cell phones present in the building. If more cell phones enter the base station's service area, the resource requirements increase accordingly, but at the same time, the base station has access to more cell phones to scale its services.

In this paper, we propose a Mobile Cellular Cloud Architecture (MoCCA) – that aims at building cloud computing systems using smart mobile phones. Unlike traditional clouds that depend on co-located servers connected via a local area network, we propose building a cloud using loosely-connected micro-servers – small footprint code that runs on a cell phone – connected via a wireless link to a base station. The contributions of this paper are as follows: (1) Identification of challenges in building mobile clouds; (2) Introduction of a churn tolerant cloud computing architecture that uses microservers running on mobile phones as compute nodes and facilitates building an autonomic cloud over wireless cellular networks. (3) Demonstration using an analytic performance model that a cellular cloud is indeed feasible. The rest of this paper is organized as follows: Section 2 presents issues and challenges in building a cloud with mobile phones. The details of MoCCA are presented in Section 3. Section 4 includes the performance of a set of numerical applications suitable for cloud computing. We discuss the state of the art/related work in mobile cloud computing in Section 5 followed by the conclusions.

2 Challenges

A mobile cloud introduces a set of new challenges in terms of operation and security of the cloud components, as well as the data objects. The challenges mainly arise from the fundamental differences between the network topology and node characteristics of a wired data center network and a cellular mobile network. In this section, we discuss the operational and security related challenges in designing a mobile cloud.

2.1 Operational Issues

Connectivity: The biggest issue facing a mobile cloud is connectivity. Data centers are built using high-speed wired networks, operating under the control of the cloud service provider. In contrast, mobile clouds, built using small-footprint server code running on mobile phones, do not have fast and high-bandwidth connectivity. Unlike a data center connected via a local area network, mobile phones use the star topology where a large number of phones are connected to a base station. The phones do not communicate among themselves directly rather they connect through the base station. This difference in topology implies that we need solutions that allow cloud protocols to deal with frequent network disruptions.

Computational limitation: Mobile phones do not possess powerful computational capability. Unlike traditional cloud servers with multi-core, high-speed processors, mobile phones may not be as resource rich. Hence, mobile clouds must be designed with the computational limitations in mind. The task assigned to each of the mobile cloud nodes should be such that it is within its resource constraints such as CPU, memory, bandwidth and the energy.

Churn: A mobile cloud needs to consider churn in a large scale. Since mobile phone users are not limited to a single location and may move in and out of range of a given cellular base station, there is a high chance that a mobile cloud node will drop out, run out of power, or have the server process interrupted or terminated. The mobile cloud architecture and protocols must be designed with enough redundancy to handle such large scale churn.

Energy: Mobile phones are power limited devices. Computations and data transfer will drain the battery. Hence, the amount of data assigned to each mobile node needs to be small enough to be feasible to compute and transfer efficiently.

User incentive: Since the nodes in the mobile cloud are mobile phones owned by different people, we need to provide an incentive model for the users to allow the use of their phones. Specifically, the users need to have a compensation model where they will benefit from letting their unused phones be a part of a mobile cloud. One possibility can be that the cell phone company may provide a discount or monetary incentive in return of the use of phone CPU cycles and data bandwidth. Since, cell phones are typically unused most of the day, especially during off-peak hours. So, the cell phone owners can benefit financially by letting the phone company utilize the unused cell CPU cycles.

2.2 Security and Privacy Issues

We mention these issues here rather briefly to appraise readers to security challenges that mobile cloud computing faces. But, security issues are beyond the scope of this paper.

Confidentiality and privacy: Unlike traditional cloud servers, the individual mobile phones in a mobile cloud are owned by different people, some of whom may not be trustworthy. Hence, it is vital to ensure that attackers cannot gather confidential information by pretending to be a legitimate user. Even if the adversary takes over a phone, or infiltrates the cloud, she should learn only small pieces of data (that was sent to her phone) and nothing else about the whole computation. Also, the cloud server code running on the phone should not gather sensitive personal information from the user's phone. Another challenge is to ensure the security of data in transit – in mobile clouds, most of the data transfer will occur over wireless links, which brings in the possibility of eavesdropping attacks.

Integrity: Mobile clouds need to ensure the integrity of data as well as computations. Dishonest users can try to benefit by misreporting the result of a computation (perhaps by sending random results without performing the actual computation). We must also ensure that only the intended data processing functions (either standard or user-defined) were applied to produce the results.

Forensics: A mobile cloud must have strong support for digital forensics. As the nodes are no longer owned by the service provider, the risk of attacks is higher. Hence, we need schemes for logging, provenance management, and forensics to identify misbehavior.

3 MOCCA: Mobile Cellular Cloud Architecture

To build clouds using mobile phones, we introduce the *Mobile Cellular Cloud Architecture (MoCCA)*. On a high-level, MoCCA is based on a small-footprint microserver code running on mobile phones. These microservers are co-ordinated by a management process running in the base station. In this section, we present an overview of MoCCA which assumes GSM cellular network as the basis for the mobile cloud. We begin by discussing background information on GSM network topology.

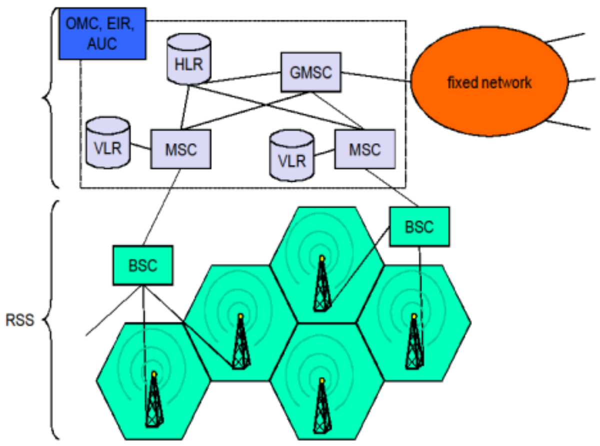

Figure 1 Architecure of a GSM cellular system [17].

3.1 Background

GSM is a widely used narrowband digital cellular system belonging to the 2nd generation (G) and has further evolved to GPRS (General Packet Radio Service – 2.5G), and UMTS (Universal Mobile Telecommunications System – 3G) systems. A functional architecture of GSM [17] is shown in Figure 1. Here, each base station is shown as a hexagon which serves N number of mobile stations (MS). A base station controller (BSC) controls multiple base stations and interfaces to a mobiles switching center (MSC). For location and mobility management of mobile devices Home Location (HLR) and Visitor Location (VLR) Registers are used [17].

3.2 MoCCA Overview

In MoCCA, the mobile cloud comprises of smart phones, base stations (BS), base station controllers (BSC), and mobile switching centers (MSC). A small mobile cloud can be formed out of mobile stations served by a single base station. A medium cloud can be constructed out of phones served by multiple base stations within the control of the same BSC. A larger cloud can be formed out of coverage area supported by multiple BSCs that are under the control of one MSC. A much larger cloud can still be constructed at the PSTN (Public Switch Telephone Network) level that includes multiple MSCs. Because of the hierarchical nature of cellular networks, MoCCA is inherently scalable.

We make the following assumptions in order to develop MoCCA: (1) The BSC has the ability to partition the client initiated workload into multiple smaller chunks that can be processed by a multiple of smart phones. (2) The BSC has a list of smart phones that are willing to participate in sharing the workload assigned to the cloud. (3) Using the control plane (existing signaling architecture), a BSC can allow a phone to join or leave the cloud as and when necessary. However, for better performance, it is reasonable to expect that a phone participates in a cloud for a minimum amount of time that is agreed upon during the initialization. (4) Phones have required software to compute the results, or it could be downloaded from the base station or any other appropriate facility. (6) After completion of the assigned job, the mobile phone informs the base station that the job is completed using a message on a control channel and transmits the result on one of the data channels assigned by the base station. (7) The BSC is responsible for verifying the correctness of the results received from the multiple phones.

3.3 Operational Model

MoCCA uses a dataflow model of cloud computing, such as MapReduce [5]. In such data processing systems, each node performs a mapping or reducing function. Results from one stage are fed into the next stage. We consider data-parallel computations.

To submit a job to MoCCA, a cloud client contacts a BSC and sends it the data and the specification of the code that should operate on the data (e.g., map and reduce functions). The BSC then contacts the MoCCA microservers running on the mobile phones, through the MoCCA manager software running on the base stations.

MoCCA microserver: Each mobile phone runs a MoCCA microserver process that handles communication with the base station and management of the server function and data. A controller component in the microsever manages the communication with the MoCCA manager, retrieval and deployment of function code and data, maintenance of accounting information, and the return of the results to the BS. The code is executed in a sandbox (similar to Java Applets), and is prevented from accessing any resource outside the sandbox. This ensures that any malicious code will not be able to read any personal information stored on the phone.

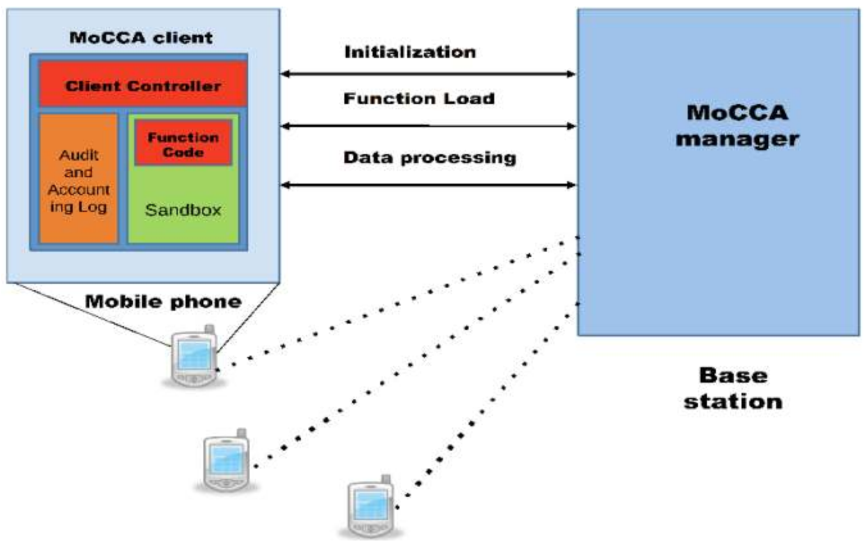

Figure 2 Operational model of MoCCA.

Operation: On a high level, the communication between a mobile phone and the base station happens in three phases.

1. *Initialization phase*: When a mobile wants to join the MoCCA framework, it sends a join request to the base station controller. The BSC then initiates an authentication protocol that is based on challenge and response methodology. To further strengthen this protocol, after the phone is authenticated, the BS and the MS can perform a Diffie–Hellman key exchange to establish a cryptographic session key. This key is used to encrypt any future communication between the cell phone and the base station. In this phase, the MS may also negotiate the type of service it is willing to provide to the BSC including the amounts of data it will process and the number of sessions for which it will remain active. Accounting for the service is also initiated in this phase.
2. *Function load phase*: In this phase, the MoCCA manager in the BS sends function code to the phone. The code is placed in the microserver sandbox.
3. *Data processing phase*: In this phase, the BSC sends one or more sets of data to the phone via BS. On receiving a data object, the MoCCA microserver places the data object in the sandbox, and invokes the previously loaded function to operate on the data object.

After the end of each computation, the microserver controller sends the result and accounting information back to the base station. The operational model of MoCCA is shown in Figure 2.

3.4 Data and Control Plane Operations

In MoCCA, all cloud computing messages whether related to control or data plane are handled on designated control and data channels thereby suggesting minimum or no modifications to existing GSM system.

3.5 Performance and Reliability

Bandwidth: The bandwidth provided by emerging cellular systems for data applications has been increasing over the years due to the emergence of bandwidth hungry mobile applications such as mobile web-surfing and multimedia. At the present time GSM/Edge systems provide a bandwidth of 384 Kbps which increases to 2 Mbps for UMTS/DECT systems. Emerging 4 G systems such as LTE are planned to have downlink bandwidth of 84–168 Mbps [24].

Energy: For cloud applications running on mobile phones, energy consumption in computation and communication (transmission and reception) is of paramount importance. For example, in GSM based applications running on Android G1 smart phones, the cost of transmission of a 200 byte data packet is 4.67 Joules while the cost of reception is 2.05 Joules [2]. The energy cost of computation is several orders of magnitude less than wireless communications. The average energy capacity of Android G1 battery is approximately 15000–20000 Joules. These results justify our hypothesis that mobile phones can indeed be appropriate platforms to act as servers for mobile clouds.

it Frequent Connection/Disconnection: In cellular communication, the probability of connection impairments leading to eventual disconnection is very high when a cloud server node is mobile. Also there is another important difference that is related to mobile cloud. In a cellular voice or data applications at the termination of the session, no data transmission takes place. But in the case of mobile cloud, the phone is responsible for sending back the computed results to the original BSC completing the task. There are several issues that arise in this context such as:

1. A MS acting as a cloud server is mobile within the coverage area of the serving base station and does not cross the cell boundary; so no handoffs take place. If there is a disconnection while computed results are being transferred to the base station, or software is being downloaded from the base station, or input data is being transmitted by the base station, then these items (text and/or data) need to be retransmitted by the base station or phone using the existing mechanisms in place for cellular data communications or as specified in standards.
2. The second scenario arises when a phone acting as a cloud server moves to a different cell resulting in a handoff or series of handoffs. We assume computed results will reach the originating base station following the handoff trail.

3.6 Reassembly and Correctness

The base station controller will be assigned the task of reassembling the results from different smart phones and ensuring that the transmission errors on the wireless channels have not introduced any errors in the computed results. The assumptions that we make in MoCCA is that cellular network architecture and governing specifications do not change to accommodate mobile cloud computing. Cellular networks implement forward error correction to deal with the channel errors and this will be first line of defence with regard to transmission errors. In addition in MoCCA, we introduce another feature to ensure correctness of the received results based on the theory of Triple Modular Redundancy (TMR) proposed by Von Neumann [18]. In TMR identical computations are assigned to three smart phones outputs of which are compared at the base station controller through a voter which chooses the correct result. A TMR configuration is shown in Figure 3. Here the duplication with output comparison is considered as an error detection technique.

The availability of the third copy of the computation provides enough redundant information to allow error masking in any one of the three copies. This is accomplished by means of a majority (two-out-of-three) vote on the three copies of the computed results. The reliability of the results of this configuration can be given as

$$R = R_v \times (R_m^2 + 3R_m^2(1 - R_m)), \tag{1}$$

where R_v and R_m are the reliabilities of the voter and a single copy of triplicated computations. The concept of triple modular redundancy can be extended to include N copies with majority voting at the base station con-

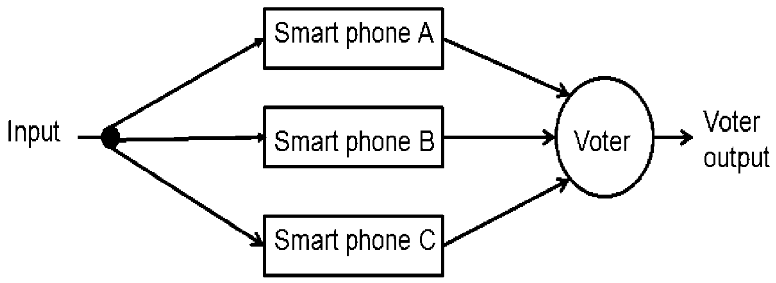

Figure 3 Reassembly and correctness in MoCCA.

troller. Equation (1) can be extended to N modular redundancy scheme to Equation (2), if higher degree of reliability in computed results is required.

$$R = R_v \sum_{i=0}^{N/2} \binom{N}{i} R_m^{(N-1)} + (1 - R_m)'. \tag{2}$$

As part of the research agenda, we will examine what is the optimal redundancy that is required to guarantee the correctness of the results and its energy and transmission cost implications. This work is planned for the future.

4 Performance Evaluation

MoCCA is a GSM based cloud whose performance we evaluate here but it can act as a reference when cloud architectures based on 3G and 4G networks that support application bandwidths in excess of 20 Mbps are evaluated. With higher bandwidths such systems should perform better than GSM for cloud applications.

We have computed the performance of a mobile cellular cloud under the following scenarios consisting of different workloads, configurations, probability of blocking, probability of handoffs under different vehicular speeds, and energy consumptions.

Workloads: We have mainly chosen compute bound applications, such as: (1) Common matrix computations e.g. inversion, eigenvalues and eigenvectors, determinant, Fast Fourier transform, and Cholesky decomposition. (2) Sorting of large arrays and linear regression. (3) Fibonacci number calculations, etc.

Configurations: Same computations are performed under three configurations: (i) a single mobile station (no redundancy), (b) Double modular redundancy, (c) Triple modular redundancy.

Probability of blocking: Computations have been repeated for different probabilities.

Mobility: We have also computed resultswith different handoff rates, the probability of handoffs, and probability of handoff droppings under different mobility models.

Energy: For all these scenarios, we have computed the energy consumption for transmission, reception and computations to find the total energy expenditure related to cloud computation. However, due to space limitation, we are not able to include all the results in this paper.

For the analysis purpose, we consider one GSM cell that has 53 channels for uplink and 53 channels for downlink transmissions. A GSM frame consists of 8 time slots each of which has duration of 577 micro-seconds in which 114 bits of data can be carried. We assume a MS to be equipped with a 500 MHz processor with 256 Mbytes of memory that runs a MS windows operating system. Our main reason for choosing these parameters is that several compute bound applications running on similar hardware have been bench-marked [22] and quite a few commercial mobile phones available on the market have specifications within the range.

In the analysis that we present here, we have assumed that a base station keeps a fraction of channels for incoming voice traffic and the remaining for cloud applications. In order to compute the capacity of the cloud for the compute bound applications, we make use of the trunking theory [17]. We define λ to be the average number of session request rate and μ the service rate for each MS which gives us a traffic intensity of $A_u = \lambda\mu$ Erlangs, with the probability of blocking, p_m:

$$p_m = \frac{\frac{1}{m!}\left(\frac{\lambda}{\mu}\right)^m}{1 + \sum_{n=1}^{m} \frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n} . \tag{3}$$

Here m is the number of channels and n is the number of MSs in the coverage area. In the mobile cloud architecture the originator of the call is the

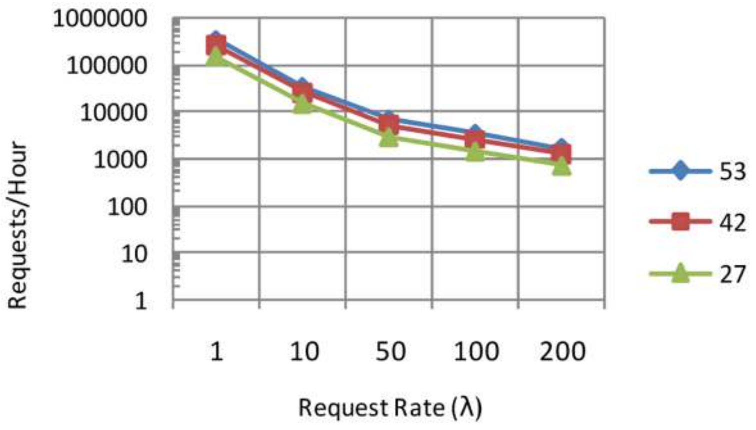

Figure 4 Cholesky decomposition.

base station and the call is terminated on mobile phones serving the agreed upon application. In the next sections we describe the cloud performance for several applications computed assuming no redundancy in the hardware.

4.1 Application 1 – Cholesky Decomposition

We assume a 100 by 100 matrix to be factorized on the mobile devices. Figure 4 plots the total number of such requests completed by the MSs when they utilize 53 channels (Blue Curve), 42 channels (Red Curve), or 27 channels (Green Curve) with respect to the request rates assigned to each MS by the base station. For example with 53 channels, with probability of blocking of 0.005, a maximum number of Cholesky decomposition that can be completed in one hour are 343,583 on participating mobile devices. In Figure 3, the x-axis depicts request rates assigned to mobile devices per hour by the base station. For example, when the base station assigns 1 request/hr to mobile devices using 27 channels, the maximum number of requests completed is 147000 per hour. But this number changes to 14700 per hour when each participating device is completing 10 requests per hour giving us the total request completions of ($10 \times 14700 = 147000$) which is still the same.

So in this case the number of participating devices becomes 14700. The number of participating devices reduces to 1470 when each device is able to complete 100 requests per hour. So depending upon the number of participating devices and the number of channels, the base station can distribute the workload according to application needs.

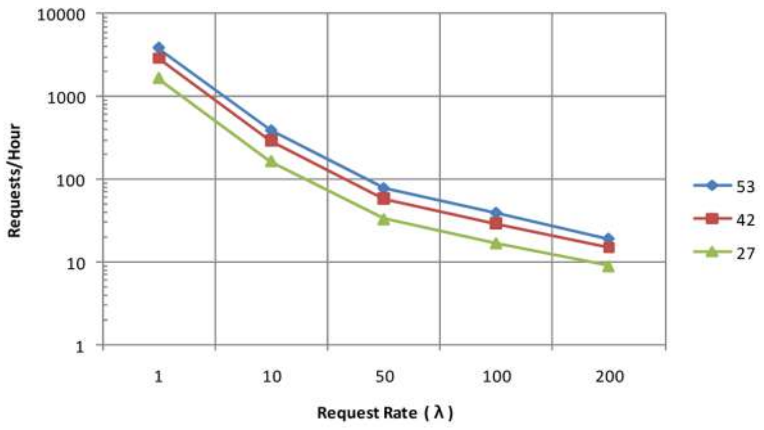

Figure 5 FFT computation – one iteration on 900,000 samples.

4.2 Application 2 – Fast Fourier Transform using S

The second compute bound application is related to the one iteration of Fast Fourier Transform (FFT) computation of 900, 000 data samples using the S [23] program package. We assume that S is running on the mobile. We use the same set of channels for this experiment. Figure 4 depicts the request completion rate. With 42 channels, 2905 instances of the applications can be run in an hour with $p_m = 0.005$.

This application requires a channel holding time of 36.5 seconds.

4.3 Application I Capacity

Figure 6 shows the capacity of application 1 with the probability of blocking for different number of channels which increases with the increase in the probability of blocking (P_m).

4.4 Handoff Performance

For the computation of the handoff probability, we make an assumption that the cloud session originated in the cell with $1/\mu$ as the duration of the session which is represented by a random variable T_n that has exponential distribution given by $f_n(t) = \text{Prob}(T_n \leq t) = 1 - \mu e^{\mu t}$. We assume the mobile is moving within the cell with a constant velocity of V Kilo-meter/hr in a cell that has a radius r.

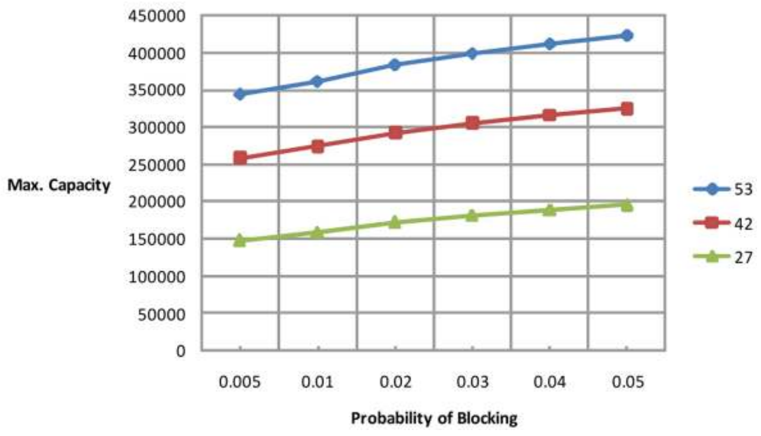

Figure 6 Probability of blocking.

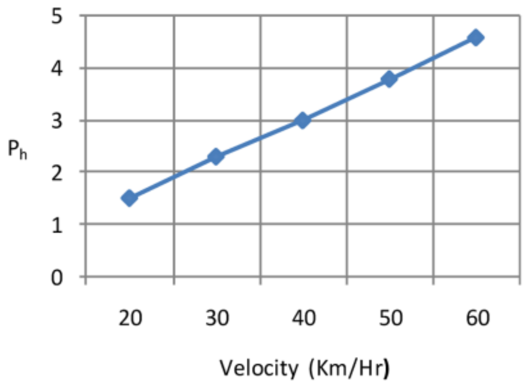

Figure 7 Handoff probability.

Assuming a circular cell for simplicity, we can write the rate of cell crossing $\eta = 2\text{V}/\pi r$ which implies that mobile is dwelling in the cell for $1/\eta$ duration. Assuming cell dwell time to be exponentially distributed, it can be represented by the probability density function $f_h = \eta e^{-\eta T_h}$, where T_h is the random variable representing the dwell time. With these parameters, we can write the expression for the probability of handoffs:

$$P_h = \text{Prob}(T_h > T_h) = \int_0^{\infty} \eta e^{-\eta T_h} \left[\int_{T_h}^{\infty} \mu e^{-\mu t} dt \right] dT_h = \frac{\eta}{(\eta + \mu)}. \quad (4)$$

Figure 7 depicts the variation of handoff probability with respect to the velocity of the mobile for a cell radius of 10 Kilometers for application 2. The probability of handoff varies from 1.5% at velocity of 20 Km/hr to 4.5% with velocity of 60 Km/hr. The handoff probability results suggest that for a typical cloud application the probability of handoffs is quite low at modest vehicular speeds.

4.5 Energy

A mobile phone participating in a cloud computing application expends energy in data computation, reception and transmission. Even after emergence of hardware and software features that help conserve the energy expenditure such as (a) turning the clock rate down when display is off, (b) dynamic voltage scaling, and (c) dynamic frequency scaling etc. saving power in mobile phones is still remains a major challenge. Recent power measurements that have been made on a mobile phone running numerical linear algebraic algorithms which are part of LINPACK benchmark [25] lead to the following observations and conclusions: (a) The energy expenditure per byte reduces with increasing packet sizes. For example a 64 byte packet may consume 31.25 milli-watts (mW) per byte when compared with a 512 byte packet consuming 3.906 mW per byte. (b) The energy expenditure per byte reduces when multiple packets are transmitted at the same time for example transmissions of 2, 8, 16, or 128 packets per second. The corresponding power consumption varies from 31.25 mW per byte to 0.49 mW per byte.

As stated earlier in the paper that the energy cost of computations is several order of magnitude less than the cost of transmissions or receptions, we only include energy results for the transmission and the reception for a few applications considered in the paper. Figure 8 depicts the variation of energy consumption with respect to packet size when only one packet per second is transmitted. The energy expenditure in Joules (J) varies from 770 J for a 64 byte long packet to 48.20 J for a 1024 byte packet when a Cholesky decomposition is performed on a 1000 $\times$ 1000 symmetric positive definite matrix Figure 6 shows the results for the energy consumption for application I with respect to number of packet size when different number of packets per second (pps) are transmitted. As one can notice that the energy consumption for a 128 byte packet is 12 J when 64 pps are transmitted and it reduces to 6.9 J when 128 pps are transmitted. The percent reduction in energy consumption is more than 50% which is significant. Figure 9 shows the energy expenditure in data reception with respect to packet size for Cholesky decomposition.

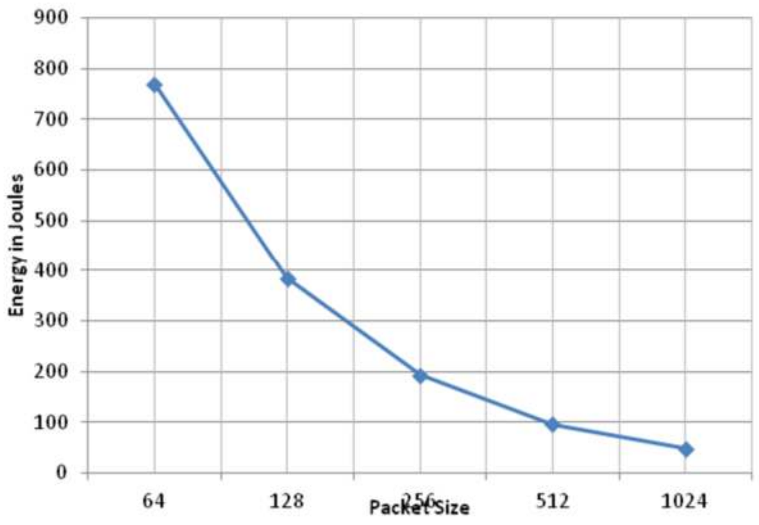

Figure 8 Energy consumption in transmission.

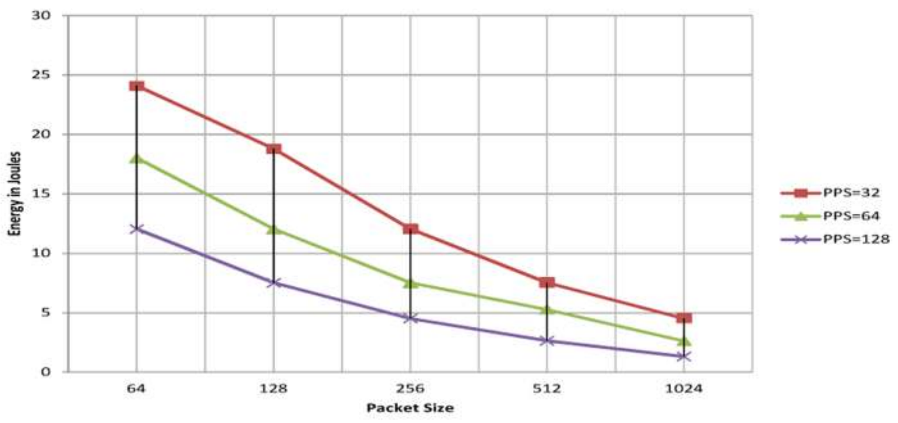

Figure 9 Energy consumption in transmission.

Using a 64 byte packet size the total energy expended in reception is 270 J when 2 pps are received and it reduces to 5.18 J for a 512 byte packet with a reception of 16 pps.

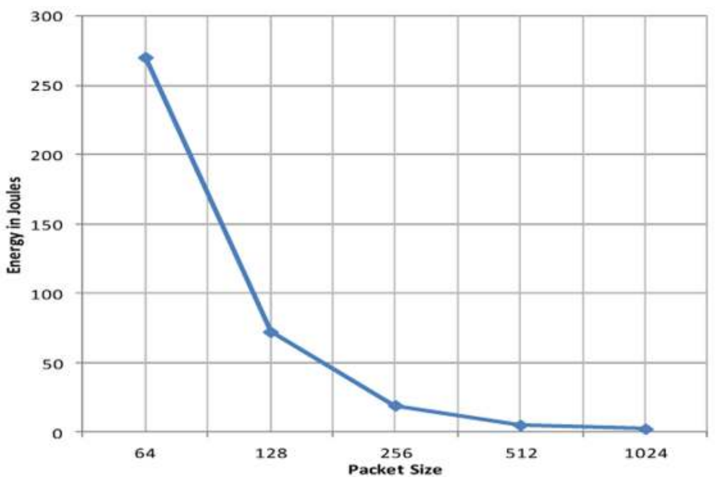

Figure 10 Energy consumption in receptiopn.

5 Related Work

In this section, we review the current research in the mobile cloud computing area. Ideas which appear similar to MoCCA have been proposed for sensor networks [25] and mobile grids [14]. But MoCCA is very different from sensor network based grids, mobile grids, or any other peer-to-peer network based grids. The important differences are: (1) Each MS belongs to a different owner, (2) Each MS is battery powered, (3) MSs are mobile, and therefore the nodes forming a cloud keep changing with respect to time, and as a consequence have (4) Dynamic cloud topology. Since the mobile devices do not directly communicate with each other i.e. there is no peer-to-peer communication, the impact of dynamic topology is not an issue in MoCCA, but this can be a major issue in clouds that rely on peer-to-peer networking. Mobile grid [1, 4, 11, 19] studies different aspects of mobile grid computing such as problem of multiple connects and disconnects, collaboration among mobile devices that are heterogeneous in terms of resources, interworking among mobile and stationery computers, integration of mobile phones to infrastructure based wireless grid. Both the wireless grid and mobile grid advocate some form of peer-to-peer networking.

WIPDroid [3] is a platform created for Droid phones [9] by integrating WIP (Web Service Initiation Protocol) for real-time service oriented communication over IP which has been mentioned as it could provide SaaS type of service as a client but not as a server which is the focus of this paper. Wireless sensor networks can constitute a valid approach to mobile cloud computing and in some ways this approach may resemble MoCCA, if we assume the sink of a sensor network to be acting like a base station of cellular networks, and the sensor field resembles the coverage area of the base station. But sensor networks and cellular networks have many profound differences e.g. the data transmission among sensor nodes is multi-hop whereas in cellular networks its only one hop. Finally, somewhat related to MoCCA are mobile data sharing systems which use distributed file systems and peer-to-peer systems such as [6, 10, 15] and mobile computing platforms such as Hyrax [11].

6 Conclusion

This paper presented MoCCA – an architecture for building mobile clouds that leverages cellular infrastructure, thus making it inexpensive and easy to deploy. The performance analysis of single GSM base station suggests that several numerical and statistical algorithms can run on large data sets in the cellular environment even with a small number of mobile phones participating only for several minutes. With the deployment of LTE that provides larger bandwidths and uses resource rich smart phones running at 1 GHz speed, mobile clouds have potential to become popular. As part of the future work, we are working on characterizing the overhead of MapReduce, additional control signaling, and energy consumption in transmission and computation. We are planning on to extend the current GSM model to a 4G LTE system that has much larger bandwidth and several attractive features for data applications.

References

[1] S. Ahuja and J. Myers. A survey on wireless grid computing. The Journal of Supercomputing, 37:1, 2006.

[2] Z. Chen, Energy-efficient Information Collection and Dissemination in Wireless Sensor Networks. PhD thesis, University of Michigan, 2009.

[3] W. Chou and L. Li. WIPdroid – A two-way web services and real-time communication enabled mobile computing platform for distributed services computing. In IEEE International Conference on Services Computing, Vol. 2, pp. 205–212. IEEE, 2008.

[4] D. Chu and M. Humphrey. Mobile OGSI. NET: Grid Computing on Mobile Devices. In 5th IEEE/ACM International Workshop on Grid Computing, pp. 182–191. IEEE Computer Society, 2004.

[5] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. Communications of the ACM, 51(1):107–113, 2008.

[6] G. Ding and B. Bhargava. Peer-to-peer file-sharing over mobile ad hoc networks. In Proceedings Second IEEE Annual Conference on Pervasive Computing and Communications, pp. 104–108. IEEE, 2004.
[7] Gartner Inc. Competitive Landscape: Mobile Devices, Worldwide, 1Q10. Online at http://www.gartner.com, May 2010.
[8] International Telecommunication Union, ITU Corporate Annual Report. Online at http://bit.ly/aJSfQs, 2009.
[9] O. Kharif. A warm welcome for Android. BusinessWeek, January 2008.
[10] C. Lindemann and O. Waldhorst. A distributed search service for peer-to-peer file sharing in mobile applications. In Proceedings 2nd IEEE International Conference on Peer-to-Peer Computing, pp. 73–80, 2002.
[11] E. Marinelli. Hyrax: Cloud Computing in Mobile Devices using MapReduce. Master's thesis, Carnegie Mellon University, 2009.
[12] J. McKenderick. The $80 data center: Cheap computing or head in the cloud? Online at http://zd.net.bBooxt, November 2007.
[13] L. McKnight, J. Howison, and S. Bradner. Wireless grids – Distributed resource sharing by mobile, nomadic, and fixed devices. IEEE Internet Computing 8(4):24–31, 2004.
[14] P. Mell and T. Grance. The NIST Definition of Cloud Computing. Version 15, 10-7-09, National Institute of Standards and Technology, 2009.
[15] N. Michalakis and D. Kalofonos. Designing an NFS-based mobile distributed file system for ephemeral sharing in proximity networks. In Proceedings 4th Workshop on Applications and Services in Wireless Networks (ASWN), pp. 225–231. IEEE, 2005.
[16] R. Miller. Microsoft's Windows Azure Cloud Container. Online at http://bit.ly/cenSFw, November 2009.
[17] M. Mouly and M. Pautet. The GSM system for mobile communications. Telecom Publishing, 1992.
[18] J. Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components. Automata studies (1956), 43–98.
[19] N. Palmer, R. Kemp, T. Kielmann, and H. Bal. Ibis for mobility: Solving challenges of mobile computing using grid techniques. In Proceedings 10th Workshop on Mobile Computing Systems and Applications, p. 17. ACM, 2009.
[20] M. Patterson, D. Costello, P. Grimm, and M. Loeffler. Data center TCO. A comparison of high-density and low-density spaces. Thermes, 2007.
[21] A. Mishra. Security and Quality of Service in Ad hoc Wireless Networks. Cambridge University Press, 2008.
[22] Stephen Stenhouse's benchmark, version 2, www.sciviews.org/benchmark/benchmark1.html
[23] Richard Becker et al. The New S Language. Wadsworth & Brooks/Cole, 1988.
[24] Erik Dahlman et al. 4GLTE/LTE Advanced for Mobile Broadband. Academic Press, 2011.

Biographies

Amitabh Mishra is a faculty in the Information Security Institute of Johns Hopkins University in Baltimore, Maryland. His current research is in the area of cloud computing, data analytics, dynamic spectrum management, and data network security and forensics. In the past he has worked on the cross-layer design optimization of sensor networking protocols, media access control algorithms for cellular-ad hoc inter-working, systems for critical infrastructure protection, and intrusion detection in mobile ad hoc networks. His research has been sponsored by NSA, DARPA, NSF, NASA, Raytheon, BAE, APL, and US Army. In the past, he was associate professor of computer engineering at Virginia Tech and a member of technical staff with Lucent Technologies – Bell Laboratories in Naperville, Illinois. His has worked on architecture and performance of communication applications running on 5ESS switch. GPRS, CDMA2000 and UMTS were a few of the areas he worked on while with Bell Laboratories. He received his B. Eng. and M. Tech. degrees in Electrical Engineering from Government Engineering College, Jabalpur and Indian Institute of Technology, Kharagpur in 1973, and 1975 respectively. He obtained his M. Eng. and Ph. D. in 1982, and 1985 respectively also in Electrical Engineering from McGill University, and a MS in Computer Science in 1996 from the University of Illinois at Urbana-Champaign. Dr. Mishra is a senior member of IEEE, a member of ACM, and SIAM. He is author of the book *Security and Quality of Service in Wireless Ad hoc Networks*, published by Cambridge University Press (2007). He is a technical editor of *IEEE Communications Magazine*.

Gerald Masson is a Professor (Emeritus) of Computer Science at Johns Hopkins University, Baltimore, Maryland. He was the founding director of Johns Hopkins University Information Security Institute and chair of the computer science department. His research interests are in fault tolerant computing, real-time error monitoring of hardware and software, inter-connection networks, and computer-communications. He is a Fellow of IEEE.