COMBAT: mobile-Cloud-based cOmpute/coMmunications infrastructure for BATtlefield applications

Tolga Soyata[a], Rajani Muraleedharan[a], Jonathan Langdon[b], Colin Funai[a], Scott Ames[c], Minseok Kwon[d], Wendi Heinzelman[a]

[a]Dept. of Electrical and Computer Engineering, University of Rochester, Rochester, NY 14627;
[b]Dept. of Biomedical Engineering, University of Rochester, Rochester, NY 14627;
[c]Dept. of Computer Science, University of Rochester, Rochester, NY 14627;
[d]Dept. of Computer Science, Rochester Institute of Technology, Rochester, NY 14623

ABSTRACT

The amount of data processed annually over the Internet has crossed the zetabyte boundary, yet this *Big Data* cannot be efficiently processed or stored using today's mobile devices. Parallel to this explosive growth in data, a substantial increase in mobile compute-capability and the advances in cloud computing have brought the state-of-the-art in mobile-cloud computing to an inflection point, where the right architecture may allow mobile devices to run applications utilizing Big Data and intensive computing. In this paper, we propose the MObile Cloud-based Hybrid Architecture (MOCHA), which formulates a solution to permit mobile-cloud computing applications such as object recognition in the battlefield by introducing a mid-stage compute- and storage-layer, called the *cloudlet*. MOCHA is built on the key observation that many mobile-cloud applications have the following characteristics: 1) they are compute-intensive, requiring the compute-power of a supercomputer, and 2) they use Big Data, requiring a communications link to cloud-based database sources in near-real-time. In this paper, we describe the operation of MOCHA in battlefield applications, by formulating the aforementioned mobile and cloudlet to be housed within a soldier's vest and inside a military vehicle, respectively, and enabling access to the cloud through high latency satellite links. We provide simulations using the traditional mobile-cloud approach as well as utilizing MOCHA with a mid-stage cloudlet to quantify the utility of this architecture. We show that the MOCHA platform for mobile-cloud computing promises a future for critical battlefield applications that access Big Data, which is currently not possible using existing technology.

Keywords: Cloud computing, mobile devices, cloudlet, Big Data, object recognition

1. INTRODUCTION

In military operations, object recognition applications play an important role. For example, military personnel could wear night vision goggles with object recognition capabilities to identify key targets swiftly and with high accuracy. The two crucial aspects of these types of object recognition applications are accuracy in identifying the targets and response-time or latency. While the accuracy is controlled in part by the quality of the object recognition algorithms, such processing inherently entails a considerable amount of computation with large, if not enormous, database support. This leads to the question of whether we can provide a system that supports the performance of object recognition in the battlefield, allowing troops to identify target objects with high accuracy in real-time or near-real-time.

Despite rapid advances in mobile and computing technologies, many troops today still rely on traditional equipment including maps, compasses, radio and GPS devices as a guide during warfare.[1] One reason is that time is critical in warfare, where even a second delay may jeopardize the life of a serviceman or even endanger national security. Many recent advances in technologies enable mobile devices to be an additional accessory for the 21st century soldier.[1] However, several technical challenges still exist in the development and deployment of such systems. First, the sheer volume of information, i.e., Big Data, accessed by mobile devices increases computing

Email: {soyata,rmuralee,funai,wheinzel}@ece.rochester.edu, jonathan.langdon@rochester.edu, sames@cs.rochester.edu, jmk@cs.rit.edu

time significantly. Second, mobile devices in the battlefield are connected to servers remotely over satellite links whose latency is far from negligible. Third, mobile devices cannot choose to utilize only locally available resources to avoid all of these problems since mobile devices have limited computing and storage resources. These limitations may cause slow computation and substantially increased response time, and thus adversely affect the quality or speed of responses for threat detection.

As a solution, we propose a mobile-cloud computing architecture called MOCHA (MObile Cloud-based Hybrid Architecture) that is well-suited for military applications handling Big Data. In this paper, we consider specifically object recognition applications, especially in the context of the battlezone. To run such applications using existing systems, a mobile device would take a picture of an observed object, send a request along with this image to a server running in the cloud, and the server would then compare this image with images from its database and return a matched result back to the mobile device. These images and results are likely to be transferred over satellite links with long latency, as discussed earlier. In an effort to reduce this delay, in MOCHA we introduce a middle box called a *cloudlet* between the mobile device and the cloud for data pre-processing and caching. Using the cloudlet, the mobile would send the picture to the cloudlet over a high speed link, and then the cloudlet would perform either some or all of the processing. If the cloudlet could not complete the processing itself (or if it lacked the necessary data from a large database), it would send the remaining computation requests to the cloud through a satellite link. The cloud would respond to the cloudlet, which would then send the final response back to the mobile device. In this paper, we use simulations to evaluate the benefit of the MOCHA architecture compared with having the mobile communicate with cloud servers directly. Our aim is to investigate the effectiveness of MOCHA, and to recommend how to minimize overall response time given resource limitations. In our scenarios, the cloudlet is located within close proximity to the troops, such as in a tank or a helicopter near the squad.

Our contributions are summarized as follows:

1. We formulate a mobile-cloudlet-cloud architecture called MOCHA for running computationally intensive applications, such as object recognition applications, from mobile devices with the goal of minimal response time.
2. We provide insight into the benefits of using the cloudlet under military environments. We compare the performances of the situations when we use the cloudlet (MOCHA) and when we do not have access to the cloudlet, and we study under which conditions the system benefits most or least from the cloudlet.

The rest of the paper is organized as follows. In Section 2, we describe our MOCHA architecture, providing details of each component. In Section 3 we develop an approach to reducing the overall response time using an optimized algorithm and run simulations to show the benefits of this optimized approach as well as the benefits of the MOCHA architecture. In Section 4, we provide an overview of related work, and we summarize our conclusions and future work in Section 5.

Table 1. Comparison of the normalized capabilities of a mobile device, a cloudlet, and the cloud

	Mobile Device	**Home Cloudlet**	**Enterprise Cloudlet**	**Cloud**
Compute	1	$100 - 10K$	$10K - 100K$	$10K - 100M$
Memory	1	$100 - 100$	$100 - 1000$	$1000 - 10M$
Storage	1	100	$10K$	$100K - 1M$
Latency	1	10	10	1000

2. THE MOCHA ARCHITECTURE

Despite the incessant increase in the compute capabilities of mobile devices driven by consumer demand, the same increase is also observed in the capabilities of desktop computers, approximately at the same rate. This is due to the architectural and technological improvements being applied to mobile platforms as well as desktop

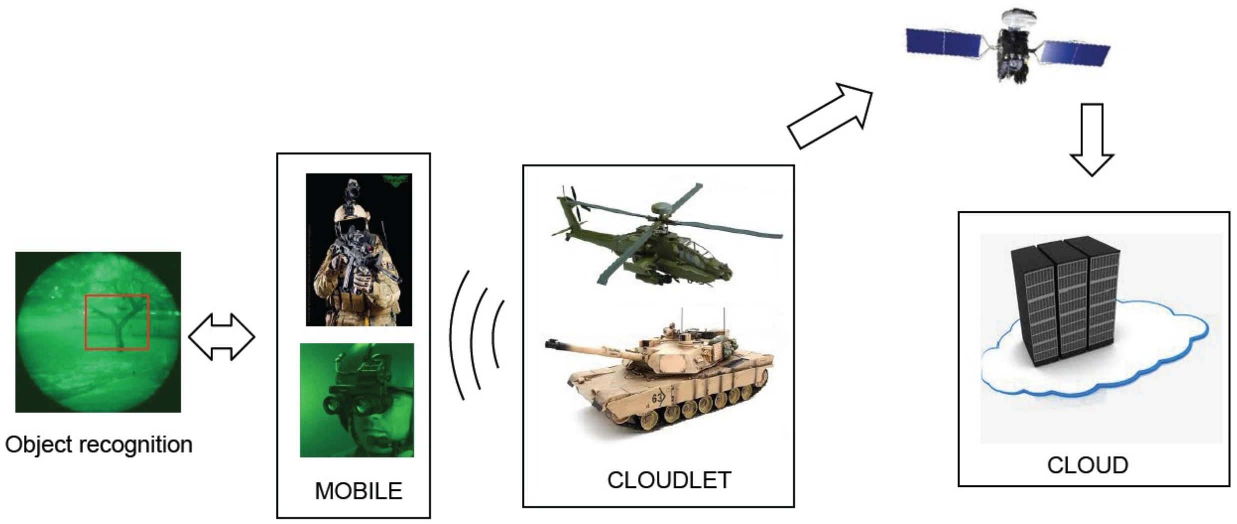

Figure 1. The MOCHA Architecture: soldiers' mobile devices interact with the cloudlet in the tank or helicopter and the cloud via network connections including satellites in the battlefield.

platforms simultaneously, driven by an equally healthy demand for compute-power in both mobile and desktop platforms. The approximate relative computational parameters for mobile devices and a large cloud provider, such as Amazon AWS,[2] as well as both a home-based and an enterprise *cloudlet*, are shown in Table 1. This table shows the large gap between the processing capabilities of the mobile, the cloudlet and the cloud, both in terms of processing power and network latencies. This gap is expected to stay the same for the foreseeable future due to the two- or three-orders-of-magnitude difference in the energy source of the mobile and desktop platforms unless game-changing advances can be made in battery technology. Assuming a linear relationship between the energy source and the compute-capability of computational devices, this implies that the large gaps shown in Table 1 will not close in the foreseeable future.

This promises a rich set of applications that can use mobile devices as the input source, with the acquired data sent through the Internet to the cloud for processing and the results sent back to and displayed on the mobile device. Synergistic coupling of the mobile and the seemingly infinite-compute-power of the cloud in this way is theoretically an excellent solution and might be the most cost-effective option for applications that are not sensitive to latency. However, as described in the Microsoft MAUI project,[3] some applications might never be feasible from mobile devices, due to the high latency of the mobile-cloud connection. Adding a *cloudlet*, a local device that provides 100 to 1000 times higher computational power than the mobile with minimal latencies, creates possibilities for running latency sensitive and computationally-intensive applications such as battlefield object recognition from a mobile device.

As a solution to massively parallelizable mobile-cloud applications (e.g., battlefield object recognition), we propose using the MOCHA architecture illustrated in Figure 1. We envision the cloudlet to be the middle layer located very close to the mobile (i.e., in a nearby tank or helicopter), providing three or four orders of magnitude higher compute-power than the mobile, embedded in the soldiers' goggles. The cloud is accessed through a high-latency satellite link either from the mobile or from the cloudlet, when the cloudlet cannot complete the required tasks alone.

2.1 Mobile Device

We propose the mobile layer of our MOCHA architecture to be a soldier's night-vision goggles, or other sensory input devices located on the soldier. While some pre-processing is possible on the mobile, higher computation at the mobile layer implies higher battery located on the soldier. Therefore, we propose the main task of our mobile device to be to capture and send data (e.g., images) to the cloudlet for pre-processing. If the cloud was closely located so the communication latency was negligible, the captured data could be sent to the cloud directly, bypassing the cloudlet. However, due to the multi-second latency of the satellite communication links,[4] an intermediate stage buffer is necessary, using the cloudlet. While certain threat-detection functions can tolerate multi-second delays, many of them require response times less than a second, making it necessary to use the intermediate cloudlet layer to permit utilization of advanced timing optimization and communications algorithms.[5–7]

To quantify the compute-demands of a real-time object recognition application, we have run an experimental object recognition algorithm and detected an object from a 1000-entry object database. Using the widely-accepted OpenCV[8] library, this computation took 12 s on an INTEL i7-960 computer, running a single thread. Increasing the object dictionary will only increase the runtime. In this specific scenario, we can quantify the power-demand by multiplying the 12 s by the 130 W that the CPU consumes: a total of 1,560 J total energy, which would have reduced to around 250 J for a multi-threaded version of the same program, due to the 4C/8T architecture of the i7. Even at the 10 frame-per-second object detection rate, the power consumption requirement would be 2,500 J per second (i.e., 2,500 W) for real-time continuous object detection. Such a power source is feasible inside a tank, but placing it on the soldier would mean a 25 kg. (55 lbs.) battery lasting only for one to two hours using state-of-the-art Li-Ion technology.[9] Hence, we need another solution to enable this application running from the mobile on the soldier.

2.2 Cloudlet

The cloudlet is the middle-stage layer with significantly higher compute, communications, and storage capabilities as compared to the mobile. We analyze the effect of each one of these separately.

2.2.1 Compute Capability

The cloudlet is conceptualized to have the capability of massively parallel processing (MPP)[10, 11] where a few ten thousand threads are actively executing as opposed to plain parallel processing (PP) where only a few hundred simultaneous threads are active. Graphics Processing Units (GPUs) are designed specifically in support of MPP utilizing commodity GPU hardware[11] and provide an excellent solution for the type of applications that are being described in this paper, with a significantly better Watts-per-GFLOPS metric than CPUs. We envision liquid-cooled ruggedized rackmount cases inside the tank, containing ten to twenty GPUs (e.g., Nvidia GTX 590 with a 2.5 TFLOPS compute capacity[12] and a 350 W power consumption each). By today's state-of-the-art standards, such an array of GPUs would consume a few kW, and would provide close to 20 to 30 TFLOPS of peak compute-power, forming a ruggedized super-computer. Improved compute power allows the cloudlet to run much more sophisticaed algorithms to determine the optimum paths of the computation between mobile and cloudlet and cloudlet and cloud (e.g., synchronous optimization[13, 14]), as well as pre-processing of the input data before it is dispatched to the cloud. Such a high compute capability will also allow the cloudlet to perform most of the functions itself without requiring dispatching to the cloud, allowing it to act as the *compute-buffer*.

Such a compute-power could only be assumed to be available on the soldier in 2025+ at today's rate of growth, making it necessary to place this layer inside a unit that can provide such power, such as the tank. In addition to the insensitivity to the weight issue, a tank also uses fossil-fuel-fired power generation which in itself has a few orders-of-magnitude advantage for power density. The power density issue is exacerbated due to the demand for real-time encryption and decryption of the data that is constantly being transferred between the mobile and the cloudlet (or between mobile and the cloud), also placing a significant demand on the compute-capability of the mobile, making it even more difficult for the mobile to cope with the object recognition task itself.

2.2.2 Storage Capability

In addition to the compute-capability of the cloudlet, another drastic advantage can help many database-driven applications: storage-capability. While a mobile device can be conceptualized to have a storage of hundreds of GB, processing this data requires compute-power, diminishing the usefulness of having large amounts of storage in the mobile, without the availability of matching compute-power. Furthermore, large amounts of storage (e.g., multiple TB) is only available in the form of hard drives made out of mechanical parts, which are not very practical to locate on the soldier. Vibration-insensitive flash or other non-volatile storage is, therefore, necessary (e.g., Solid State Disk - SSD) with implied limits of less than 1 TB. The cloudlet, on the other hand, can contain hundreds of these units, implying a two- to three-order-of-magnitude higher storage. This has a direct correlation to the capability of the cloudlet to cache a portion of the Big Data necessary for processing (e.g., object dictionary). With this *storage-buffer* role, the cloudlet can significantly reduce the number of database elements that have to be transferred from/to the cloud, thereby having a significant impact on the overall performance.

2.2.3 Communications Capability

One final advantage of the cloudlet is the ability to relieve the mobile from the unreliability and location-sensitivity of the satellite communication. Since satellite communication is very location-sensitive, placing this responsibility on the cloudlet allows the cloudlet to use much higher communications energy to efficiently communicate with the satellite with a much higher reliability, and potentially collaboratively use the other cloudlets for communications acceleration. This allows the soldier to access its compute-resources (whether it is the cloudlet or the cloud) through much more reliable links (e.g., secure WiFi) and run the application reliably since the cloudlet acts as the *communications buffer*.

We assume each cloudlet to transfer control to other cloudlets when one is down during the highly-fluid battlefield environment. We envision intelligent algorithms running on the mobile layer, continuously monitoring the available cloudlet resources and picking one (or more) that is available with the highest amount of compute-power. In the unfortunate case of zero available cloudlet resources, the mobile resorts to direct communication with the cloud layer at a drastically reduced response-time, but with full functionality and security maintained.

Table 2. Characteristic of Satellite Communication

	Globalstar	**O3b Networks**	**Wideband Global SATCOM**
Earth Orbit	*LEO*	*MEO*	*GEO*
Constellation	48	8	6
Data Rate	64 Kbps	1.25 Gbps	2.1 − 3.6 Gbps
Latency (seconds)	0.04	0.125	0.24
Coverage	120 Countries	+/ − 45 degrees of latitude	−

2.3 Cloud

In our system, we use the term "Cloud" to refer to the vast compute-resources available outside the battlefield, accessible through a satellite communication link. For battlefield applications that can tolerate the high latencies, such as pre-examination of the battlefield for certain threats, the cloudlet layer provides diminished utility and the cloud may be accessed directly. However, even if time delay is not a problem, there are advantages to going through the cloudlet to access the cloud rather than directly from the mobile device. For example, the cloudlet can perform more advanced task partitioning algorithms than the mobile, and it can more easily ensure reliable data transfer to the cloud, saving energy on the mobile. For either method (going directly from the mobile or through the cloudlet), the pre-processed and encrypted data packets can be sent to data-centers located outside the battlefield via satellite links for processing.

A list of the existing satellite links are tabulated in Table 2. The Globalstar second-generation satellites launched into LEO in 2011 support a throughput of 64 Kbps with less than 0.04 s round trip time (rtt);[15] the O3b satellites, with an expected throughput of 1.25 Gbps and an expected round trip latency of 0.125 s, are anticipated to launch into MEO in 2013;[16] and the Wideband Global SATCOM (WGS) satellites launched into GEO in 2007 provide the US Department of Defense's highest-capacity satellite communication links, supporting data throughput of 2.1-3.6 Gbps with a round trip latency of 0.24 s.[17] While MEO permits communications with the cloud efficiently, LEO has a very low throughput, making it only useful for applications with small data sizes. GEO, on the other hand can transfer data very efficiently, albeit at much higher latencies, limiting its usefulness for applications requiring very low response times. Important aspects of the cloud for consideration are:

- Since the cloud amasses a virtually infinite compute-capability, it is the only option for certain tasks that are intractable even for the cloudlet. Example applications are the ones requiring a compute power in excess of the 100 TFLOPS range.

- Even if a single application (such as the aforementioned object recognition) does not demand such a high compute-power, running the same application for ten to twenty soldiers on the cloudlet simultaneously might. Therefore, it is necessary to run intelligent algorithms on the battlefield to send non-latency-sensitive data to the cloud to avoid burdening the cloudlet with unnecessary processing.
- Although the tank will contain a significantly higher power density than a soldier, performing such computations in the tank will consume its resources (e.g., fuel), which has an important implication in the battlefield, further emphasizing the importance to send all tasks to the cloud when they can tolerate high latencies.
- Despite the high communication latency to reach the cloud, the compute-capability of the cloud is near infinite. This implies an inflection point, where the runtime reduction from the cloud might actually surpass the latency when the overall compute time is calculated. This will hold true for extremely high compute-demand applications, or the case of significant unavailability in the cloudlet compute power. This could happen due to damaged tanks, or damaged cloudlet units inside the tanks.
- While the cloudlet can cache a large portion of the database that is necessary for processing (e.g., object dictionary), it is limited by the state-of-the-art storage technologies. While it is possible for the cloudlets to cache around 100TB to 300TB of the database, the cloud can very well contain a multi-PB (peta-byte) database, making it necessary to access the cloud when the object being searched is not in the database of the cloudlet.

3. EVALUATION OF THE BENEFITS OF MOCHA

When performing any application using cloud (or distributed) resources, there are three main concerns: (1) accuracy of the result, (2) latency or response-time in obtaining the result, and (3) cost. While the accuracy of the result depends on the computation performed, the response-time depends on both the processing time at the various servers as well as the communication latency among the different computing devices and back to the end point. Cost depends on the type of cloud resources (public or private) being used. Clearly, these three performance metrics can be traded-off against each other. For example, a less sophisticated computation algorithm can be run to reduce response-time at the cost of reduced accuracy, or more servers can be used to perform the computation to reduce response-time for the same accuracy at an increased cost.

In this section, we analyze the impact of our MOCHA architecture in a computationally-intensive battlefield application that requires a fast response-time with limited communication resources, but has no cost constraint. Battlefield applications rely on satellite communication links to transfer data to the cloud, where the throughput and latency of these satellite links vary based on how the satellite orbits the earth (e.g., low earth orbit (LEO), medium earth orbit (MEO), geostationary earth orbit (GEO) and high earth orbit (HEO)[18, 19]). The detailed characteristic of these satellite links are shown in Table 2.

3.1 Mathematical Formulation

In our simulations, we consider three different scenarios for performing the computationally-intensive object recognition application using LEO and GEO satellite links in a battlefield scenario.

(i) Global Cloud: in this scenario, the mobile sends a 20 kB image through a satellite link to one or multiple cloud servers (C), where the processing is performed, and the results are sent back to the mobile. We consider the throughput and latency for LEO and GEO satellite links as described in Table 2. In this scenario, the response time of the application can be determined by

$$T_{total}^{M\leftrightarrow C} = t_{communication}^{M\leftrightarrow C} + t_{compute}^{C} \tag{1}$$

where $T_{total}^{M\leftrightarrow C}$ is the application response time determined by the sum of the total communication latency ($t_{communication}^{M\leftrightarrow C}$) and the total compute time at the cloud ($t_{compute}^{C}$). The former can be broken into three seperate components as shown below

$$t_{communication}^{M\leftrightarrow C} = rtt^{M\leftrightarrow SR} + t_{transfer}^{M\leftrightarrow SR} + t_{transfer}^{SR\leftrightarrow C} \tag{2}$$

where round-trip-time ($rtt^{M\leftrightarrow SR}$) is the amount of time it takes to initiate and complete the communication between the mobile (M) and the satellite receiver (SR) as described in Section 2.3. Once the communication has been initiated, the amount of time it takes to transfer the data is calculated as

$$\begin{aligned} t^{M\leftrightarrow SR}_{transfer} &= \frac{DataSize}{R^{M\leftrightarrow SR}} \\ t^{SR\leftrightarrow C}_{transfer} &= \frac{DataSize}{R^{SR\leftrightarrow C}} \end{aligned} \tag{3}$$

where $DataSize$ is the amount of the data being transferred through the channel and $R^{M\leftrightarrow SR}$ and $R^{SR\leftrightarrow C}$ are the mobile-to-SR and SR-to-cloud channel data rates, respectively. The total computation time ($t^C_{compute}$) is the sum of two separate components as follows

$$t^C_{compute} = t^C_{pre-process} + t^C_{post-process} \tag{4}$$

where $t^C_{pre-process}$ is the initial pre-processing step of the application, common to most image processing tasks (e.g., SIFT feature extraction[20]), and $t^C_{post-process}$ is the actual processing time of the information using the output from the pre-processing step (e.g., object recognition[21]). In scenario (i), we assume that both pre- and post-processing are performed at the cloud due to the compute-intensity of both tasks being too high for the mobile alone. To gain intuition into the capabilities of the three different satellite options (i.e., LEO, MEO, and GEO) shown in Table 2, the total communication latency for transferring 20kB of data is calculated as follows

$$\begin{aligned} t^{M\leftrightarrow LEO}_{communication} &= rtt^{M\leftrightarrow LEO} + \frac{20KB}{64Kbps} = 0.040 + 2.5 = 2.54s \\ t^{M\leftrightarrow MEO}_{communication} &= rtt^{M\leftrightarrow MEO} + \frac{20KB}{1.25Gbps} = 0.125 + 0.0001 = 0.125s \\ t^{M\leftrightarrow GEO}_{communication} &= rtt^{M\leftrightarrow GEO} + \frac{20KB}{3.6Gbps} = 0.240 + 0.00002 = 0.24s \end{aligned}$$

This clearly shows the large impact of rtt for MEO and GEO, for small data sizes. It is clear from Equation 2 that to see the benefits of the higher-end MEO and GEO links, the data sizes should be in the MB range or higher. Equation 2, however, does not take into account the variations in the data rate due to the unpredictability of the satellite communications. The latency for transfering the 20 kB image from the satellite receiver to the cloud server (i.e., $t^{SR\leftrightarrow C}_{transfer}$) can vary from 0 s (assuming the cloud server is at the satellite receiver) to 0.2 s when these effects are included.

(ii) Cloudlet: in this scenario, the mobile sends the 20 kB image to the cloudlet through an 802.11n link whose throughput is 100 Mbps and whose latency is 0.01 s.[22] The cloudlet performs the object recognition computation, and the results are sent back to the mobile. Following the same terminology as Equation 1, we can define the expanded equation

$$T^{M\leftrightarrow CL}_{total} = rtt^{M\leftrightarrow CL} + t^{M\leftrightarrow CL}_{transfer} + t^{CL}_{pre} + t^{CL}_{post} = 10ms + \frac{20kB}{100Mbps} + t^{CL}_{pre} + t^{CL}_{post} = 11.6ms + t^{CL}_{pre} + t^{CL}_{post} \tag{5}$$

where the pre- and post-processing times are denoted as t_{pre} and t_{post}. As is apparent from Equation 5, the response time of the application ($T^{M\leftrightarrow CL}_{total}$) in scenario (ii) is dominated by the pre- and post-processing compute steps (i.e., $T^{M\leftrightarrow CL}_{total} \approx t^{CL}_{pre} + t^{CL}_{post}$). Considering the two to three orders-of-magnitude difference between the compute capabilities of the mobile and the cloudlet, Equation 5 suggests that the response time will only degrade to below an acceptable level if the algorithm being run on the cloudlet is computationally intensive. Otherwise, the cloudlet will be able to perform both the pre- and post-processing without the necessity of accessing the cloud (assuming that all of the required database is readily available in the cloudlet memory). Following our description in Section 2.2.1, the cloudlet can act as the *compute buffer* by performing the pre- and post-processing, which would be intractable tasks for the mobile alone.

(iii) Cloudlet+Global Cloud: in this scenario, the mobile again sends the 20 kB image through a high-speed, low-latency link (802.11n) to the cloudlet, where the pre-processing is done, but the post-processing can be offloaded from the cloudlet to the global cloud through satellite links. In this case, since the cloudlet is able to preprocess the image, we assume that only 10 kB of data must be transferred to the cloud through the satellite link, but the latency for transferring the 10 kB data from the satellite receiver to the cloud server in this case will vary from 0 s to 0.1 s. Using the terminology from the previous scenarios, the total response time is

$$\begin{aligned} T_{total}^{M\leftrightarrow CL\leftrightarrow C} &= rtt^{M\leftrightarrow CL} + rtt^{CL\leftrightarrow SR} + t_{transfer}^{M\leftrightarrow CL} + t_{transfer}^{CL\leftrightarrow SR} + t_{transfer}^{SR\leftrightarrow C} + t_{pre}^{CL} + t_{post}^{C} \\ T_{total}^{M\leftrightarrow CL\leftrightarrow C} &\approx rtt^{CL\leftrightarrow SR} + t_{transfer}^{CL\leftrightarrow SR} + t_{transfer}^{SR\leftrightarrow C} + t_{post}^{C} \end{aligned} \tag{6}$$

where the approximation in Equation 6 is possible due to the significantly lower values of $rtt^{M\leftrightarrow CL}$ and $t_{transfer}^{M\leftrightarrow CL}$ and the ability for the cloudlet to perform the pre-processing in an amount of time that is significantly less than the desired response time of the application (i.e., $t_{pre}^{CL} \ll T_{total}$). As mentioned in Section 2.2, the cloudlet acts as the compute buffer by performing the pre-processing and as the communications buffer by removing the 10kB additional transfer burden from the $CL \leftrightarrow C$ channel.

3.2 Task Partitioning

In each of these scenarios, we assume an *embarrassingly parallel* application, which can be partitioned into a fixed number of independent and identical *tasks* that must be performed by one or more computational units. All tasks must be complete before the final result can be sent back to the mobile. We consider two task partitioning algorithms, which we call *fixed* and *greedy*. In the fixed approach, tasks are evenly distributed among available cloud resources (cloudlet and/or cloud servers) for processing. The overall response time is based on the time taken for the last response to be returned. In the greedy approach, the cloud resources (cloudlet and/or cloud servers) are ordered by their (known) response times to complete a task, and the first task is assigned to the cloud server that can complete this task with the minimal amount of time. We then continue assigning tasks in a sequential order to the cloud server/cloudlet that takes minimal time to complete the task (note that this may be the same cloud server as given prior tasks if this provides the minimal overall response time to complete the task). We continue partitioning tasks using a greedy approach to select the cloudlet and/or the cloud servers, and the overall response time is again calculated based on the time taken for the last response to be returned.

3.3 Evaluation of Different Scenarios

In our first set of simulations, the performances of the fixed and greedy approaches for the three different scenarios are compared using Monte Carlo simulations, for an application that is broken into five independent and identical tasks that are distributed among the available cloud and cloudlet resources. We assume that the time for the cloudlet and each cloud server to process a task is a uniform random variable between 0.01 s and 0.1 s. The time taken to perform the task partitioning algorithm at the mobile device or at the cloudlet, and the buffer time taken by the mobile device to send the image are ignored.

Figure 2 shows the average and 95% confidence bounds for the overall response time for the Global Cloud, Cloudlet and Cloudlet+Global Cloud scenarios using LEO and GEO links and both the greedy and the fixed task partitioning algorithms (note that the Cloudlet scenario doesn't perform task partitioning since all tasks are performed at the cloudlet). The availability of the Global Cloud is based on satellite coverage, and in the case the Global Cloud is not available, these results show that the Cloudlet scenario can provide a reasonable response time. Additionally, it is clear that the greedy approach greatly outperforms the fixed approach in task partitioning, reducing the overall average response time (across all three scenarios). The greedy approach requires knowledge of the latencies and the processing times for the cloud servers. This may be difficult to obtain in practice (especially for the Global Cloud scenario where the mobile would need to collect this information). However, if estimates of latencies and processing times are available, the greedy approach can provide a great improvement in response-time, which is critical for many battlefield applications.

With our next set of simulations, we aim to understand the performance of the different scenarios as the computational load increases. Thus, we analyze the performance of the Global Cloud, Cloudlet and Cloudlet+Global Cloud scenarios when we vary the number of tasks that must be completed from 1 to 10. In this simulation set, the processing time per task of all the cloud servers and the cloudlet is once again a uniform random variable between 0.01 s and 0.1 s, and we assume that 5 cloud servers are available. Figure 3 shows that the Cloudlet

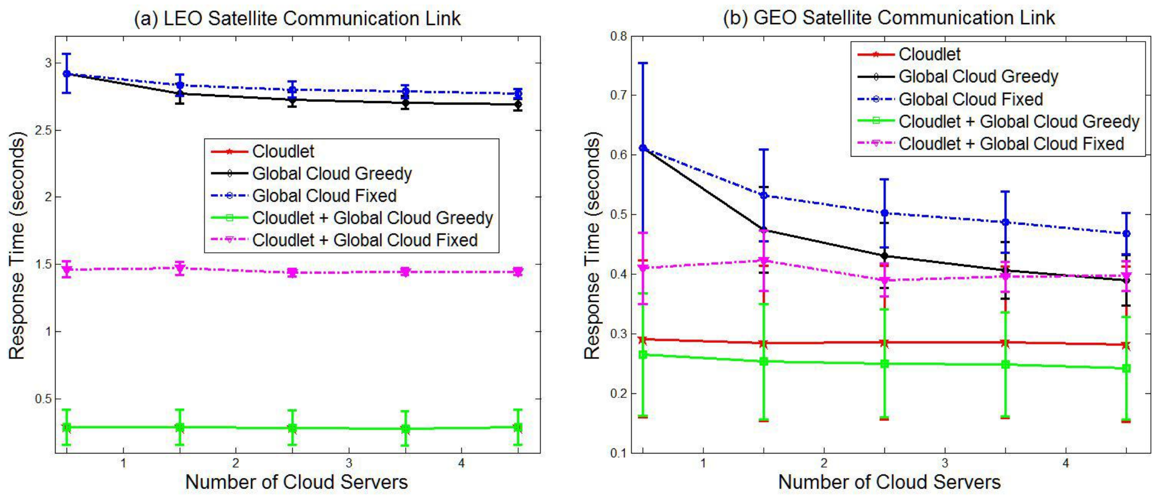

Figure 2. Simulated response times as the number of cloud servers increases [note: the Y-axis is scaled in (a) and (b) to clearly show the performances of LEO and GEO].

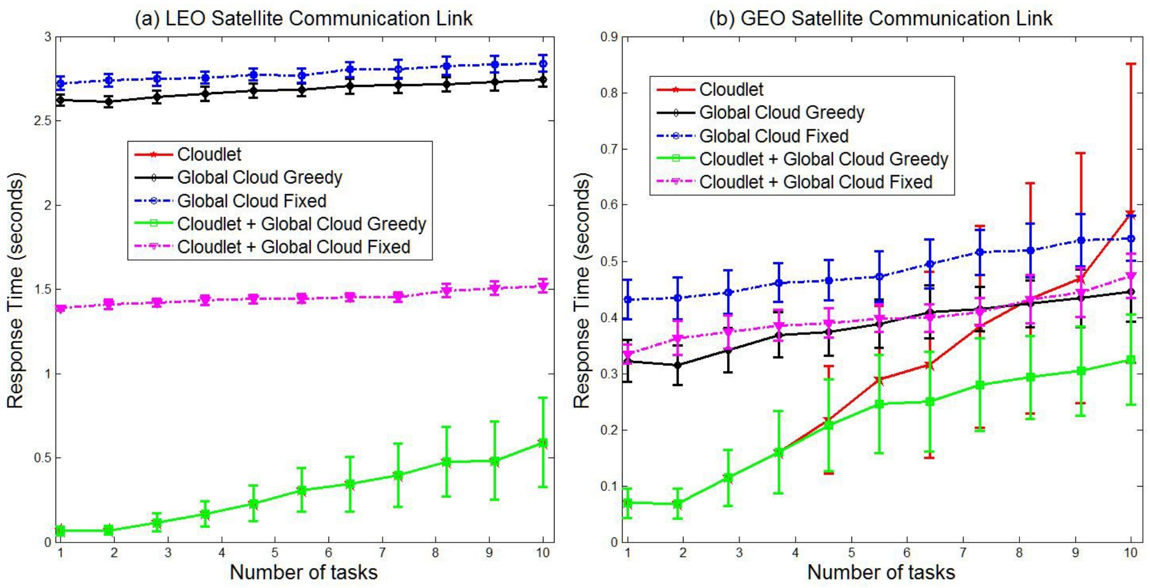

Figure 3. Simulated response times as the number of tasks increases [note: the Y-axis is scaled in (a) and (b) to clearly show the performances of LEO and GEO].

and Cloudlet+Global Cloud scenarios again provide great benefit in terms of response time, but the benefits diminish as the amount of computation (number of tasks) increase for GEO satellite links, since the long latency links become less relevant in the overall response time. This figure also shows that if the Global Cloud is not available, the application can trade-off accuracy for response time by reducing the computation (resulting in a reduction in the number of tasks to be completed).

Satellite is the primary means of communication for mobile personnel in the military, and hence communication coverage is never guaranteed. Modern satellite communication promises increased bandwidth with reduced round trip time (latency)[4, 23] . Thus, in the next set of simulations, we explore the performance of the different scenarios under varied satellite link latencies based on image sizes ranging from 1 kB to 100 kB using LEO and GEO, while the processing time of the cloudlet and the cloud servers is a uniform random variable between 0.01 s and 0.1 s, the number of tasks is set to five, and there are five cloud servers available. Figure 4 shows that the

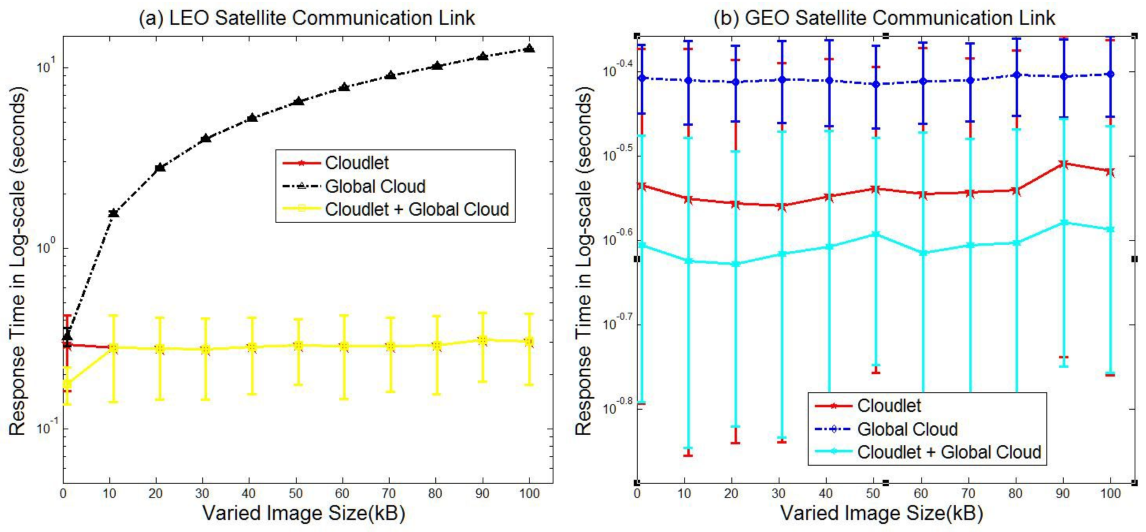

Figure 4. Simulated response times for different satellite links as the image size varies [note: the Y-axis is scaled in (a) and (b) to clearly show the performances of LEO and GEO].

Cloudlet and Cloudlet+Global Cloud are beneficial for both LEO and GEO satellite links.

In order to evaluate the speed-up provided by the cloudlet, we define $T_{M\leftrightarrow C}$ as the response time when going from the mobile to the cloud (i.e., the Global Cloud scenario); and $T_{M\leftrightarrow CL\leftrightarrow C}$ as the response time when going from the mobile to the cloudlet to the cloud. Thus, the percent speed-up in response time enabled by the cloudlet (for the greedy (*g*) and fixed (*f*) task partitioning algorithms), can be defined as:

$$\gamma^{f/g}_{M\leftrightarrow C|CL} = \frac{T^{f/g}_{M\leftrightarrow C} - T^{f/g}_{M\leftrightarrow CL\leftrightarrow C}}{T^{f/g}_{M\leftrightarrow C}} \tag{7}$$

Table 3 shows the speed-up performances for the different scenarios, summarizing the results shown in Figures 2-4. From this table, we can clearly see that the cloudlet is beneficial, providing speed-ups in response time ranging from 12% to as much as 98% for the scenarios we tested. These results show that using a cloudlet as an intermediary, as proposed in the MOCHA architecture, is beneficial under different conditions, such as varied communication latencies, computational demand and satellite links. Irrespective of the satellite coverage and latency, MOCHA guarantees a minimal response time, and the energy exploited due to heavy computation at the mobile device can also be eliminated. Also, the smart task partitioning of the greedy algorithm can enhance the cloudlet performance by choosing resources appropriately for the computation. As shown by these simulations, the benefits of MOCHA are significant, and it is well suited for computationally-demanding military applications where satellite communication is unpredictable and national/personnel security cannot be compromised.

Table 3. Percent speed-up in response time using the cloudlet for LEO and GEO satellite links and the fixed and greedy algorithms.

% Speed-up (γ)	**Number of Servers** $1 \rightarrow 5$		**Number of Tasks** $1 \rightarrow 10$		**Image Size** $1 \rightarrow 100$ (in kB)	
	Fixed	Greedy	Fixed	Greedy	Fixed	Greedy
LEO Satellite	$50 \rightarrow 48$	$90 \rightarrow 89$	$49 \rightarrow 46$	$97 \rightarrow 64$	–	$44 \rightarrow 98$
GEO Satellite	$33 \rightarrow 15$	$57 \rightarrow 38$	$23 \rightarrow 12$	$79 \rightarrow 27$	–	$35 \rightarrow 37$

4. RELATED WORK

Cloud computing has recently been considered for military, defense and national security purposes. The report titled "State of Public Sector Cloud Computing" written by Vivek Kundra,[24] the Federal Chief Information Officer, discusses the status of cloud computing technologies adopted and planned in the federal government including defense agencies. The most notable project is RACE (Rapid Access Computing Environment)[25] initiated by the DISA (Defense Information Systems Agency).[26] The RACE service provides a process for development and testing of applications to a DISA Defense Enterprise Computing Center using virtual server technology. Other projects include AEC (Army Experience Center), Forge.mil[27] again by DISA, and PSDT[28] (Personnel Services Delivery Transformation) by the United States Air Force. All of these projects use cloud computing as their basic technologies to provide computing and storage resources.

The rapid advancements in mobile technology and cloud computing have enabled many mobile cloud computing applications. Irrespective of the security concern presented in the work by Keller,[29] where face recognition and cloud computing are used to match a snapshot with a person's online identity in less than a minute, the mobile's resources and cloud computing techniques need to be exploited to benefit applications that utilize Big Data. Hyrax[30] enables smartphone applications to form a distributed computing cloud, which implements Hadoop[31] in Android phones using the MapReduce function. Similarly, Huerta-Canepa and Lee[32] developed an ad-hoc cloud computing architecture motivated by the smartphone's ubiquitousness and available resources. Chun and Maniatis[33] investigate the feasibility of dynamically partitioning application processing between weak devices (i.e., smartphones) and clouds. They formulated this partitioning problem as an optimization that minimizes execution time given resource constraints. Inspired by the fact that the data access patterns of many mobile applications depend on the current location of the user, WhereStore[34] caches cloud data on the phone based on location information using the current location and the prediction of future locations.

Energy efficiency of smart mobile devices is studied in conjunction with cloud computing in work by Miettinen.[35] RACE[36] proposes the idea that mobile phones can act as data relay nodes to augment network connectivity between servers and the battery-constrained mobile devices. Motivated by the fact that virtual data centers in the cloud partition compute power but have little control over network bandwidth partitioning, Seawall[37] explores possibilities that clients share network resources in such a way that each of them isolate themselves from others fairly. The software architecture in smartphones in support of secure personal clouds is discussed in work by Seong et al.[38] CasCap[39] is a cloud-assisted context-aware power management framework that utilizes the resources in the cloud to support secure, low-cost and efficient power management for mobile devices. Gilbert et al.[40] propose a visionary system that automatically validates how secure mobile apps are at app markets using cloud computing. Berriman et al.[41] used the Montage image mosaic engine to compare the cost and performance of processing images on the Amazon EC2 cloud and the Abe high performance cluster at the National Center for Supercomputing Applications (NCSA) to emphasize the necessity of provenance management.

5. CONCLUSIONS AND FUTURE WORK

We have presented our MOCHA platform that enables a rich new set of battlefield applications including object recognition with near-real-time responses. This platform consists of three components, namely *mobile*, *cloudlet*, and *cloud*, in which the cloudlet helps reduce the high latency of the satellite communication between the mobile and the cloud. In our context, the cloudlet is located in mobile military vehicles such as aircrafts or tanks in communication with sensory devices equipped with soldiers (*the mobile*). Our simulations show that the cloudlet indeed accelerates battlefield applications at the expense of extra hardware. Our results also indicate that intelligent algorithms for distributing tasks from the cloudlet to the cloud datacenters are crucial in order to reduce the overall response time given the long latency of satellite links. Overall, we conclude that the MOCHA architecture shows promise for battlefield applications that need to access Big Data for processing.

REFERENCES

[1] Hennigan, W., "Military opening up to hand-held devices," (2011). http://www.thenewstribune.com/2011/09/28/1842929/military-opening-up-to-hand-held.html.

[2] Amazon, "Amazon Web Services (AWS)." http://aws.amazon.com.

[3] Microsoft Corporation, "Mobile Assistance Using Infrastructure (MAUI)," (2011). http://research.microsoft.com/en-us/projects/maui/.

[4] Bilogrevic, I., [*Satellite Communications: Internet challenges and strategies for low-latency applications*], MS Project: Royal Institute of Technology, Sweden (2008). http://infoscience.epfl.ch/record/147212/files/Satellite_Communications_Internet_low-latency_Igor_Bilogrevic.pdf.

[5] Soyata, T. and Friedman, E. G., "Synchronous performance and reliability improvements in pipelined ASICs," in [*Proceedings of the IEEE ASIC Conference*], 383–390 (Sep 1994).

[6] Soyata, T., Friedman, E. G., and Mulligan, J. H., "Monotonicity constraints on path delays for efficient retiming with localized clock skew and variable register delay," in [*Proceedings of the International Symposium on Circuits and Systems*], 1748–1751 (May 1995).

[7] Soyata, T. and Friedman, E. G., "Retiming with non-zero clock skew, variable register and interconnect delay," in [*Proceedings of the IEEE Conference on Computer-Aided Design*], 234–241 (Nov 1994).

[8] Bradski, G. and Kaehler, A., [*OpenCV Computer Vision with OpenCV Library*], O'Reilly (2008).

[9] Wikipedia, "Lithium-Ion Battery," (2012). http://en.wikipedia.org/wiki/Lithium_Ion.

[10] Kirk, D. B. and Hwu, W.-M., [*Programming Massively Parallel Processors*], Morgan-Kaufmann (2010).

[11] Lindholm, E., Nickolls, J., Oberman, S., and Montrym, J., "NVIDIA TESLA: A Unified Graphics and Computing Architecture," in [*IEEE MICRO*], 39–55 (November 2008).

[12] Nvidia Corp, "GeForce GT 500 Series," (2011). http://en.wikipedia.org/wiki/GeForce_500_Series.

[13] Soyata, T., Friedman, E. G., and Mulligan, J. H., "Integration of clock skew and register delays into a retiming algorithm," in [*Proceedings of the International Symposium on Circuits and Systems*], 1483–1486 (May 1993).

[14] Soyata, T. and Friedman, E. G., "Incorporating Interconnect, Register and Clock Distribution Delays into the Retiming Process," *IEEE Transactions on Computer Aided Design of Circuits and Systems* **CAD-16**, 105–120 (Jan 1997).

[15] Selding, P., "Six Globalstar Satellites Healthy in Orbit Following Successful Soyuz Launch," (2011). http://www.spacenews.com/launch/111229-globalstar-sat-healthy-orbit.html.

[16] Gill, F., "O3b Networks," (2012). http://www.technologytimes.pk/2012/02/15/fysal-gill-rd-o3b-networks/.

[17] Pitts, T. and Fjeseth, C., "Transformational wideband communication capabilities for the warfighter," (2012).

[18] Roddy, D., [*Satellite Communications third edition*], McGraw-Hill (2001).

[19] Morton, R., "Super-secret spy satellite launched in cloud of secrecy," (2010). http://www.examiner.com/cia-in-national/super-secret-spy-satellite-launched-cloud-of-secrecy/.

[20] Wikipedia, "Scale-invariant feature transform (SIFT)," (2012). http://en.wikipedia.org/wiki/Scale-invariant_feature_transform.

[21] Viola, P. and Jones, M. J., "Robust real time face detection," in [*Second International Workshop on Statistical and Computational Theories of Vision - Modeling, Learning, Computing, and Sampling*], 1–25 (July 2001).

[22] "IEEE Standard for Information technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput," *IEEE Std 802.11n-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, and IEEE Std 802.11w-2009)* , c1 –502 (29 2009).

[23] King, M. and Riccio, M. J., "Military Satellite Communications: Then and Now," (2010). http://www.aero.org/publications/crosslink/spring2010/04.html.

[24] Kundra, V., "State of Public Sector Cloud Computing," (May 2010). http://www.cio.gov/documents/StateOfCloudComputingReport-FINALv3_508.pdf.

[25] DISA, "Rapid Access Computing Environment," (2008). http://www.disa.mil/race.

[26] Department of Defense, "Defense Information Systems Agency." http://www.disa.mil/.

[27] DISA, “Forge.mil.” http://forge.mil/.

[28] The United States Air Force, “Personnel Services Delivery Transformation.” http://www.arpc.afrc.af.mil/library/psd/index.asp.

[29] Keller, J., “The Atlantic: Cloud-Powered Facial Recognition is Terrifying,” (2011). http://www.theatlantic.com/technology/archive/2011/09/cloud-powered-facial-recognition-is-terrifying/245867/.

[30] Marinelli, E., “Hyrax: Cloud Computing on Mobile Devices using MapReduce,” Tech. Rep. CMU-CS-09-164, Carnegie Mellon University (September 2009).

[31] Hadoop Project Management Committee, “Hadoop,” (2008–present). http://hadoop.apache.org.

[32] Huerta-Canepa, G. and Lee, D., “Ad Hoc Virtual Cloud Computing Providers for Mobile Devices,” in [*Proceedings of ACM Mobile Cloud Computing and Services (MCS)*], (June 2010).

[33] Chun, B. and Maniatis, P., “Dynamically Partitioning Applications Between Weak Devices and Clouds,” in [*Proceedings of ACM Mobile Cloud Computing and Services (MCS)*], (June 2010).

[34] Stuedi, P., Mohomed, I., and Terry, D., “WhereStore: Location-based Data Storage for Mobile Devices Interacting with the Cloud,” in [*Proceedings of ACM Mobile Cloud Computing and Services (MCS)*], (June 2010).

[35] Miettinen, A. and Nurminen, J., “Energy Efficiency of Mobile Clients in Cloud Computing,” in [*Proceedings of Usenix HotCloud*], (June 2010).

[36] Jung, E., Wang, Y., Prilepov, L., Maker, F., Liu, X., and Akella, V., “User-Profile-Driven Collaborative Bandwidth Sharing on Mobile Phones,” in [*Proceedings of ACM Mobile Cloud Computing and Services (MCS)*], (June 2010).

[37] Shieh, A., Kandula, S., Greenberg, A., and Kim, C., “Seawall: Performance Isolation for Cloud Datacenter Networks,” in [*Proceedings of Usenix HotCloud*], (June 2010).

[38] Seong, S.-W., Seo, J., Nasielski, M., Sengupta, D., Hangal, S., Teh, S. K., Chu, R., Dodson, B., and Lam, M., “PrPl: A Decentralized Social Networking Infrastructure,” in [*Proceedings of ACM Mobile Cloud Computing and Services (MCS)*], (June 2010).

[39] Xiao, Y., Hui, P., Savolainen, P., and Yia-Jaaski, A., “CasCap: Cloud-assisted Context-aware Power Management for Mobile Devices,” in [*Proceedings of ACM Mobile Cloud Computing and Services (MCS)*], (June 2011).

[40] Gilbert, P., Chun, B.-G., Cox, L. P., and Jung, J., “Automated Security Validation of Mobile Apps at App Markets,” in [*Proceedings of ACM Workshop on Mobile Cloud Computing*], (June 2011).

[41] Berriman, G. B., Deelman, E., Groth, P., and Juve, G., “The application of cloud computing to the creation of image mosaics and management of their provenance,” *SPIE Conference 7740 Software and Cyberinfrastructure for Astronomy* (2010).