An Energy-Efficient Middleware for Supporting Multimedia Services in Mobile Grid Environments

Yun Huang, Shivajit Mohapatra and Nalini Venkatasubramanian
School of Information & Computer Science,
University of California, Irvine, CA 92697-3425, USA
{yunh, mopy, nalini}@ ics.uci.edu

Abstract

In this paper, we present techniques for exploiting intermittently available resources in grid infrastructures to support QoS-based multimedia applications on mobile devices. Specifically, we integrate power aware admission control, grid resource discovery, dynamic load-balancing and energy adaptation techniques to enable power deficient devices such as PDAs to run distributed multimedia applications. Our integrated solution adapts to dynamic changes in device energy consumption and unpredictable grid resource availabilities without compromising application Quality of Service (QoS). Our simulation results indicate that our power-aware grid-based approaches, not only improve QoS of mobile multimedia services, but also efficiently load-balance the grid resources.

1. Introduction

Providing support for multimedia applications on low-power mobile devices remains a significant research challenge. This is primarily due to two reasons: (i) Portable mobile devices have modest sizes and weights, and therefore inadequate resources - low CPU processing power, display capabilities, limited memory and battery lifetimes. (ii) On the other hand, multimedia applications tend to have distinctive QoS and processing requirements which makes them extremely resource-intensive. This innate conflict introduces key research challenges in the design of applications, distributed adaptation algorithms and device-level power optimizations.

The grid computing environment [1] provides an ideal setting for applying proxy based techniques to aid and improve power/performance of low-power mobile devices. Conventionally, a grid consists of an aggregation of networked computers forming a large-scale distributed system that can be exploited to solve computational and data intensive problems. However, in [4], the authors suggest the use of a proxy based clustered architecture to support mobile nodes within a grid. While traditional grid based research has focused on making grids more flexible, secure, exploring effective resource sharing techniques and addressing other issues like scheduling, co-allocation and accounting, we investigate how grid resources can be effectively utilized to support multimedia streaming to mobile low-power devices. More specifically, we try to harness the idle resources of the computational grid (to act as "proxies") to provide "energy-aware'' services to power constrained mobile hosts, while ensuring that services are not interrupted either due to intermittent availability or overloading of grid resources.

However, making the grid "power-aware'' poses several challenges. Firstly, grid resources are intermittently available; scheduling policies must take availability of grid resources into account. Secondly, effective resource allocation policies must be designed to address the performance-quality tradeoffs for the multimedia applications. Thirdly, the energy constraints of devices, limited capability of grid resources for handling multiple services and mobility of the devices necessitate the need for efficient and adaptive resource management policies. To overcome the aforementioned challenges, specifically, in this paper, we: **(1)** propose a global power-aware request-admission algorithm; **(2)** integrate proxy-based strategies into the framework for adapting to dynamic residual energy changes of the low-power devices; **(3)** develop strategies to handle variations in grid resources (e.g. grid machine failure). These techniques are integrated into our middleware based grid framework called PAGODA (Power-Aware Grid Optimizations for mobile MultimeDia Applications). In the following sections, we will discuss the PAGODA framework and present our integrated solution. Subsequently, we will present the details of our simulation model and experimental details.

2. The PAGODA Framework

The system model for our framework is as follows: A mobile client (*MC*) (e.g. PDA, laptop etc.) runs a

Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05)
0-7695-2315-3/05 $ 20.00 **IEEE**

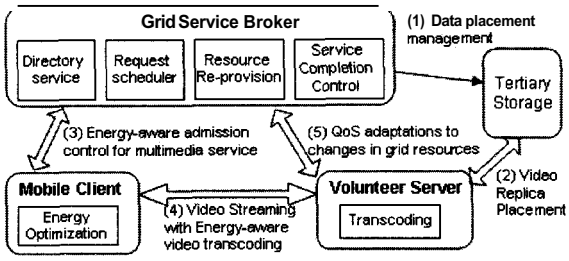


Fig.1. The PAGODA Middleware Framework

streaming multimedia application (e.g. MPEG player) while traversing a series of cells (in a cellular network), and it can receive video streams from various grid resource providers. We define a Grid ***Volunteer Server (VS)*** as a machine that participates in the grid by supplying idle resources when they are not being used, i.e. *VS*s are intermittently available. A ***VS*** in our case can be a wired workstation, server, cluster etc., which provides high capacity storage to store the multimedia data, CPU for multimedia transcoding, decompression and/or buffer memory. Additionally, the ***VS*** can be used to perform adaptive network transmission **[9]** that facilitates effective dynamic power management of device NICs (network interface cards).

In Fig.1, we show the various middleware services executing on the three system entities (i.e. Broker, the ***VS*** and the *MC*) with the arrows representing communications between the components. A ***Grid Service Broker*** forms a key component of the system and is assumed to be a high end server. The ***Broker*** is responsible for global adaptations in PAGODA. It performs the power aware admission control (i.e. it decides whether to accept or reject the request based on resource availability at the VS, the video quality requested and the residual battery capacity of the device) for incoming requests, It also determines the grid entities (*VS*s) that would serve the request over its entire duration and handles *VS* failures and service migrations within the grid.

The high-level algorithms executing on the three entities are shown in Fig **2.** For instance, the functionality of the Broker is implemented by various middleware services (in Fig 1), which include a Request Scheduler, a Resource Re-provisioning, and a Service Completion Control module to deal with ***VS*** failures. The ***Directory Service (DS)*** module in the ***Broker*** provides global state information about clients and resource availabilities at the *VS*s. Each ***VS*** downloads video replicas from a tertiary storage and performs dynamic video transcoding with adaptive network transmission. Similarly the low-power device has middleware services for monitoring residual battery energy and performing power optimizations (e.g. DVS).

When a new request is made by a MC, the ***Broker*** executes the energy-aware admission algorithm (discussed in Sec. **3**) based on the residual energy of the MC with the resource and network information retrieved from ***Directory Service (DS)***. If no scheduling solution exists, the ***Broker*** rejects the request; otherwise it identifies a set of *VS*s that can serve the request at various times and reserves the necessary resources on them. The selected *VS*s replicate video segments from a ***tertiary storage;*** the ***DS*** is updated to reflect the new allocation information subsequently. Once a request is accepted, the framework performs several dynamic adaptations (discussed in Sec. 4) based on the local device/*VS* level changes (e.g. device energy changes, ***VS*** resource availability changes) and global system wide changes (e.g. ***VS*** failures) to maintain the highest possible QoS for each request. Note that the ***Broker*** adapts only to global changes that affect the entire system. For example, when it detects a ***VS*** failure, it reschedules the services pre-assigned to that ***VS*** onto other *VS*s. The *VS*s locally adapt video streams to improve QoS based on their resource availability (e.g. network transmission bandwidth) and current residual battery on the mobile devices.

In this paper, we assume network connectivity at all

```
Grid Service Broker (){
// Rid is the request ID. VSid the ID of one VS.
 WHILE (TRUE)
  SWITCH ( Incoming Event)
    CASE NEW–REQUEST:
        Energy-aware–Admission–Control (Rid);
    CASE REQUEST– INTERRUPTED:
        Release–Preassigned–VS (Rid);
    CASE VS_RESOURCE_UPDATION:
        IF (RESOURCE–REDUCED)
            Resource–Re-Provisioning (VSid);
        ELSE
            IF (RESOURCE–INCREASED)
                Directory_Service_Updation(VSid);
    CASE VS_UNAVAILABLE :
       Service_Completion_Control(VSid);
}
```

```
Mobile Client (){
    Send request R to Brokerwith QH and QL and Er;
    IF (request accepted) {
        WHILE (in this video streaming service) {
            Receive video Streamingfrom VS;
            Send periodic residual energy information to VS;
            Update location periodicallyto directory service;
    IF (running out of battery)
        Shut down this service, and notify the current VS.
}
```

```
Grid Volunteer Server () {
    WHILE (TRUE)
        Receive new schedulefrom the Broker;
        VS_Local_Adaptations ();
}
```

g. **2** High Level Algorithms executed at the (a) ***Broker,*** (b) Mobile Client (*MC*) and (c) Volunteer Server (*VS*)

times and the middleware framework handles the protocol interactions and connection handoffs resulting out of tearing and re-establishment of connections. We are currently addressing these issues in another work and focus primarily on the energy and load balancing issues in this paper.

3. Energy-Aware Admission Control

Our adaptive strategies are based on the fact that streaming lower quality video (e.g. through proxy-based transcoding) to power deficient mobile clients results in sending lighter traffic over the network and less computation for decoding video frames and therefore less energy consumption at the mobile device. However, the lower quality directly affects user perception of video (QoS), so it is important to understand the notion of video quality for a handheld device and its implications on power consumption. Before presenting our energy-aware grid-based solutions, we introduce an ***E-Q (Energy/Quality) matrix*** (Table 1) for handheld computers (Compaq iPaq 3650), which leverage on work of [9] to identify video quality parameters (a combination of ***bit rate, frame rate and video resolution)*** that produces a user perceptible change in video quality and a noticeable shift in power consumption for handheld computers.

Note that the values provided in Table 1 will be different for different devices as well as various models of the same device. However, as profiling is a one time effort, we assume that such tables will be available for all mobile devices within the system. Therefore, in the rest of this paper, we apply table 1 to perform our energy-aware *VS* based dynamic video transcoding in PAGODA. Basically, using the ***E-Q matrix,*** we map the video quality levels Q_i to a network transmission bandwidth value NBW_i and a power cost value, and vice versa, e.g.:

$$P(Q_7) = TABLE1(EXCELLENT) = 4.37W \ .$$

Table 1. E-Q (Energy/Quality) matrix

QUALITY	Video transformation parameters	Avg. Power Windows CE (w)	Avg. Power Linux (w)
Q8 (original)	SIF, 30fps, 650Kbps	4.42	6.07
Q7 (Excellent)	SIF, 25fps, 450Kbps	4.37	5.99
Q6 (Very Good)	SIF, 25fps, 350Kbps	4.31	5.86
Q5 (Good)	HSIF, 24fps, 350Kbps	4.24	5.81
Q4 (Fair)	HSIF, 24fps, 200Kbps	4.15	5.73
Q3 (Poor)	HSIF, 24fps, 150Kbps	4.06	5.63
Q2 (Bad)	QSIF, 20fps, 150Kbps	3.95	5.5
Q1 (Terrible)	QSIF, 20fps,100kbps	3.88	5.38

A video streaming request ***R < VID, T, Q_L, Q_H, E_R, itinerary(opt)>*** specifies the required video object by ***VID,*** whole service period ***T,*** the lowest QoS level Q_L and highest QoS level Q_H , current residual energy E_R , and the mobile client's ***itinerary*** (optional). When the ***Broker*** receives a new request, it needs to decide whether to accept or reject the request. Specifically, it needs to check: whether the device has sufficient battery energy left to playback the entire duration of the requested video at the desired quality; whether it can assign a set of *VS*s to the request and ensure that the *VS*s have the necessary resources to serve the request.

The ***Broker*** runs the Energy-aware Admission Control Algorithm (EAC) algorithm (Fig **3**) to firstly determine the highest QoS level ($Q_h >= Q_L$) that the residual energy (E_R) on mobile client can support. If the residual energy on the device is insufficient even for the lowest quality required by the client, the ***EAC*** algorithm rejects the request; otherwise, it attempts to identify a set of appropriate *VS*s to schedule the request. Details of the scheduling algorithm, *Partition_Service_Period()* and *VS_Allocation(Q_h)* (Fig *3)*, have been addressed in our previous work [11], therefore will not be reiterated in this paper. Briefly, to ensure effective load balancing of the grid resources (*VS*s) and guaranteed QoS to the mobile device under current system conditions,

```
EAC ( R < Obj, T, QL, QH, ER > ){
  Qh = Rule-based_QoS_Mapping (ER , T, QL, QH);
  // if no sufficient energy for QL, then Qh = -1.
  IF (Qh != -1) // energy at least sufficient for QL
    BOOLEAN found = TRUE;
    Partition_Service_Period();
    FOR each chunk i
        IF (VS_Allocation(Qh,i) == -1)
        // current available VSs can not provide Qh,i
          NBW, = MAX(NBW of available VSs);
          IF (NBWm >= NBW.TABLE1(QL))
            VS_Allocation(Qh,i) = vsID;
            // vsID>0, VSvsID has maximum available NBW
            Qh,i = TABLE1(NBWm);
          ELSE
            found = FALSE;
     IF (found)
        ACCEPT(R);
        reserve resources for each chunk;
     ELSE
        REJECT(R) // insufficient VS resources
  ELSE
    REJECT(R) // insufficient device energy.
}
Rule-based_QoS_Mapping (ER , T, QL, QH) {
  Ph = ER / T;
  IF (Ph < TABLE1(QL)) Qh = -1;
  ELSE
      IF (Ph < TABLE1(QH)) Qh = Table1(Ph);
      ELSE Qh = QH
  RETURN Qh;
}
```

Fig 3. Energy-aware Admission Control (EAC)

Partition–Service–Period() is used to divide the whole service period into non-overlapping chunks (possibly of different sizes), which may apply knowledge of user mobility pattern (given ***itinerary).*** Subsequently, ***VS_Allocation(Q_h)*** maps each chunk to a suitable ***VS*** by choosing one ***VS*** that is lightly loaded and geographically close to the mobile client [11]. Eventually, if it is possible to find ***VSs*** for all chunks, there exists a scheduling solution; otherwise, the request is rejected.

4. Dynamic Adaptations in PAGODA

Within a grid environment, dynamic and unpredictable system changes can occur frequently (e.g. residual energy variations on the mobile device, unexpected resource fluctuations on the ***VS*** and even ***VS*** failures), which affect the request QoS, the service completion guarantees or both. It is therefore imperative that our system adapt dynamically to accommodate these changes. In this section, we identify and elaborate on our adaptation strategies.

Within PAGODA, dynamic adaptations can happen either at the ***VS***s or at the ***Broker.*** While the ***VS***s can adapt to smaller local changes, larger changes that affect the entire system require the involvement of the ***Broker.*** The ***VS***s typically adapt services to handle unpredictable changes in the residual energy of the device (e.g. due to starting/stopping of applications on device, energy optimizations on device) and sudden changes in their local resource availability. However, if a ***VS*** fails, the ***Broker*** has to reassign services previously assigned to the failed ***VS*** by re-executing the ***EAC*** algorithm for the victim requests.

Local Adaptations at Grid *VS*s: Fig. **4** illustrates the high level algorithm for ***VS*** based local adaptations in PAGODA. Given the latest residual energy feedback from the device, the ***VS*** dynamically determines the video quality to be streamed to each device; meanwhile, it must be able to handle unpredictable variations in its local resources. These conditions have been explained as two separate cases in the algorithm. When the ***VS*** receives the latest residual energy (e_r) value from the device, it determines the highest quality video that can be streamed to the device using function of ***Rule-based_QoS_Mapping (e_r, t, Q_L, Q_H)*** (in Fig **3**) based on e_r, the remaining time of the service, its local resource availability and a system specific adaptation policy. If the device does not have sufficient battery energy to support the entire duration of the current service, then the ***VS*** streams video at the lowest acceptable quality, and notifies the device about its depleted battery energy state. This implies that the device is consuming too much power (maybe due to other applications) and only *VS* based adaptations cannot complete the service. On he other hand, if we employ energy optimization echniques [9] on the MC, then our VS based adaptation will eventually result in an increase in the video stream quality.

```
VS_Local_Adaptations () {
  WHILE (TRUE)
    SWITCH (Event)
      CASE DEVICE–ENERGY–UPDATE:
        Perform video QoS adaptations based on the device
        residual energy and the available resources on the
        VS using Rule-based_QoS_Mapping (e_r) in Fig 3.
      CASE VS_RESOURCE_CHANGE:
        IF (resources increase)
          Send resource availability update to the Broker;
        IF (small resource reduction)
          Downgrade local services;
        IF (large resource reduction)
          Request broker for migrating services;
          IF (broker unable to re-schedule services)
            Service failed;
            Release resources.
}
```

Fig **4. VS** Location Adaptation Algorithm

The ***VS*** also needs to perform dynamic adaptations when its own resources reduce unpredictably (e.g. the owner of ***VS*** starts various applications). If the resource changes are small, the ***VS*** performs local adjustments to satisfy the QoS requirements of the current set of services. This might require downgrading the video quality of an existing subset of services. On the other hand, if there is a significant change in the resource availability at the ***VS*** affecting the completion of certain services, then a subset of services have to be migrated away from the ***VS.*** In this case, the ***VS*** informs the ***Broker*** for initiating the service migration. However, given system resource limitations, services that may not be schedulable on other ***VS***s result in service failures. After migrating a service, the ***VS*** re-adjusts the released resources to accommodate the remaining services in order to minimize the number of migrations. We limit (place an upper bound) the number of migrations of a single service by bounding the maximum number of migrations over the lifetime of the service.

Global Adaptations at Grid Broker: Due to the dynamic nature of the grid, ***VS*** availability itself can change unpredictably, which can result in service failures. To deal with this problem, the ***Grid Service Broker*** (in Fig 1) needs to reallocate other ***VS***s to the interrupted requests so as to fulfill their remaining service. When a specific ***VS*** becomes unavailable, the ***Broker*** retrieves information from the ***Directory Service*** about requests that are scheduled on the failed ***VS,*** and triggers the re-scheduling process for each invalidated service.

In order to reduce service failures and minimize service recovery time, we need to determine the order in which to migrate the disrupted services onto which *VS*s. To minimize service recovery time, the ***Broker*** classifies invalidated services into two categories: *(1)* services that have been started and *(2)* services are not yet started. Services in first category receive higher rescheduling priority than that of the second, whose service rescheduling can be postponed with an acceptable delay. If requests cannot be rescheduled, the ***Broker*** downgrades a number of the disrupted services to accommodate them in the remaining *VS*s; if they still are not reschedule-able even after downgrading the service, the ***Broker*** notifies the client that service fails and releases pre-allocated resources for the disrupted services on the other *VS*s.

5. Performance Evaluation

In this section, we analyze the system performance with various scheduling policies and adaptation strategies under different system configurations. The main metrics used to evaluate performance include: the system request acceptance ratio, the average QoS level of accepted services, and the number of incomplete services due to either insufficient energy on the device or reduced resources on *VS*s.

The Simulation Environment: In our simulation, we model the system environment as a cellular system with 'c' cells and 'v' *VS*s distributed within the area. Each *VS* has 's' gigabytes storage, and 'n' Mbps network transfer bandwidth. The CPU and memory resources of the *VS*s are assumed not to be the bottlenecks. We use a service ***TimeMap*** for ***VS***s to keep the information of when and how long the *VS* is available during the span of a day, and we apply three ***TimeMap*** models (i.e. *Uniform,* ***Random, Total availability)*** proposed in [113. Note that the ***TimeMap*** can be changed any time due **to** dynamic changes in *VS* availabilities. We characterize incoming multimedia requests from a ***MC*** by using a Zipfian distribution [5]. To characterize mobility of individual nodes, we use the incremental mobility model [6], where mobile hosts are distributed randomly and can move freely in a closed coverage area [11]. We set the duration of each video between 0 to '*d*' hours. Each video replica requires 's_i' gigabyte disk storage, and network transmission bandwidth that ranges from 100 kbps to 1.3 Mbps. The shortest segment of a video object can run for 'k' minutes. In our simulation, we initially set c=100, v=20, s=100GB, n=100, d=2, s_i =2GB and k=10, and the parameters are modified to simulate different configurations.

The device energy model is based on our experiments made on a typical handheld device, the Compaq iPaq ***3600*** running Windows CE. We use value **3.4** W as the power consumed when device CPU is idle, with super-bright back light and network interface card attached [9]. At the time of making the request, the average residual device energy on each device is generated using a random distribution. The Energy/Quality matrix presented in Table 1 is used to map video quality levels onto their corresponding power consumption levels. Additionally, applications are randomly started and stopped on the device to dynamically vary the energy consumption pattern with each application consuming an amount of energy varying randomly between 0.1W to 0.5W. We assume that the middleware on the device can determine the instantaneous residual energy $E_R(t)$ of the device by making appropriate system calls.

Energy-aware Admission Control: We evaluate the performance of the admission control using three different approaches: (1) no adaptation, *(2)* transcoding only and *(3)* the proposed EAC admission control algorithm (as in Fig. *3)*. With approach 1, we stream the highest QoS level to the devices (i.e. multiple quality levels are not supported at the servers). With approach *2,* we downgrade quality of service when the *VS* resources in the system are not sufficient for the highest QoS level. As seen from Fig. 5, approach 1 leads to lowest request acceptance ratio, but highest QoS provisioning for each service. However, as residual energy of the device is not considered, it also results in

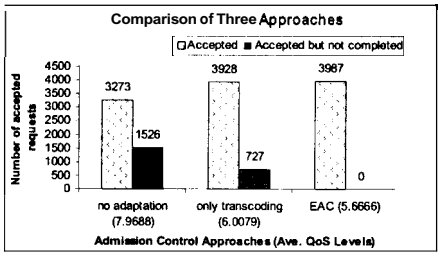
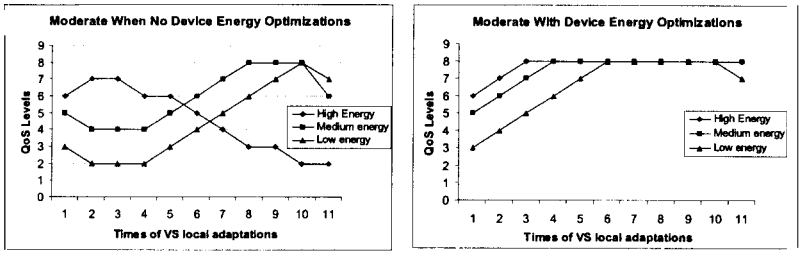


Fig 5 Three admission controls



(a) (b)

Fig 6 Impact of Device Energy Optimizations

most number of incomplete services due to insufficient device energy. Approach **2** reduces the average QoS levels for requests, and therefore accepts more requests, while reducing the number of incomplete services due to insufficient device energy. Approach **3** ***(EAC)*** takes the residual energy of the device into account, while also performing quality transcoding; thus, it accepts the highest number of requests with all accepted services completed, assuming no other dynamic changes thereafter.

Impact of Device Energy Level: Device level power optimizations (e.g. CPU voltage scaling [10], power mgmt. of network interfaces [9]) continuously provide energy gains in the mobile devices. The local adaptation at the *VS* can leverage on these energy gains to boost the QoS of the service to the device. Fig 6 shows the adaptations for three randomly selected requests (with high, medium, and low initial device energies) without (Fig 6a) and with (Fig 6b) device level energy optimizations. With device level optimizations, a constant increase in the QoS is observed and a comparatively higher QoS is maintained for all requests. (Fig 6b).

Dealing with *VS* failures: When dynamic changes to *VS* availability occur, a number of factors can affect the overall request completion ratios - the initial *TimeMap*s of each ***VS,*** and the current load etc. Given the limited space, we only present the experiment results under a ***Uniform availability TimeMap*** and ***Random*** mobility pattern [11]. In the simulation, we choose the moment when a total of 5 *VS*s are available. We gradually increase the number of simultaneous *VS* failures and observe its impact on request completion ratios. As Fig 7 illustrates, we can significantly decrease the number of requests that fail to complete due to dynamic changes in *VS* availability by applying the adaptation strategy proposed in section 4. If all *VS*s become unavailable at the same time, the adaptation strategies cannot find any available *VS* to migrate the invalidated services; similar results are therefore observed with and without adaptations. Notice that the number of services that cannot be completed increases almost linearly in the graph, indicating that the system is "load-balanced"using our approach.

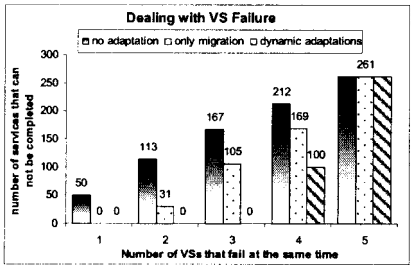


Fig 7. Dealing with *VS* Failures

6. Related Work and Conclusions

A tremendous amount of research efforts have been dedicated to achieving power savings in delivering multimedia services for mobile devices. Algorithms that address the overlap between communications and compression have been proposed in [8]. Video segmentation [3], intelligent caching and buffer management techniques, and high speed wireless transmission have been extensively researched. In addition, resource management in the grid has been addressed by several grid projects such as Globus, Condor [1], AppLes [7], and Nimrod [2], etc. In this paper, we proposed the PAGODA framework for effectively utilizing available grid resources to improve global (grid-level) and local (device-level) system performance for mobile multimedia applications. We devised an energy-aware admission control algorithm that takes into account current device capabilities and grid resource availabilities to guarantee predictable services in mobile environments. We developed a family of QoS-aware adaptation policies that handle unforeseen variations in the grid resources. In future work, we plan to address issues like location awareness, secure communications and service reliability.

7. References

[1] I. Foster, C. Kesselman, ***The Grid: Blueprint for a New Computing Infrastructure,*** book 1998.

[2] D. Abramson, J. Giddy, and L. Kotler, ***High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid?*** IPDPS ***2000.***

[3] S. Chen, et. al, ***Adaptive and lazy segmentation based proxy caching for streaming media delivery,*** NOSSDAV, 2003.

[4] T. Phan, L. Huang, C. Dulan, ***Challenge: Integrating Mobile Wireless Devices Into the Computational Grid,*** MobiCom, 2002.

[5] A. Dan and D.Sitaram. ***An online video placement policy based on bandwidth to space ration (bsr)*** . SIGMOD,1995

[6] **Z.** Haas. ***A new routing protocol for the reconfigurable wireless networks,*** Universal Personal Communications, 1997.

[7] F. Berman, R. Wolski, H. Casanova, et al. ***Adaptive Computing on the Grid Using AppLeS,*** HPCA, 1999.

[8] E. Jeannot, B. Knutsson, M. Bjorkman, ***Adaptive Online Data Compression,*** 11th IEEE HPDC, **2002.**

[9] *S*. Mohapatra, R. Cornea, et.al., ***Integrated Power Management for Video Streaming to Mobile Handheld Devices,*** ACM MM, 2003

[10] P. Pillai and K. Shin. ***Real-time dynamic voltage scaling for low-power embedded operating systems.*** SOSP ***2001.***

[11] Y. Huang, and N. Venkatasubramanian, ***Supporting Mobile Multimedia Services with Intermittently Available Grid Resources,*** HiPC 2003.