Supporting Mobile Multimedia Services with Intermittently Available Grid Resources

Yun Huang and Nalini Venkatasubramanian

Dept. of Information & Computer Science,
University of California, Irvine, CA 92697-3425, USA
{yunh, nalini}@ ics.uci.edu

Abstract. Advances in high quality digital wireless networks and differentiated services have enabled the development of mobile multimedia applications that can execute in global infrastructures. In this paper, we introduce a novel approach to supporting mobile multimedia services by effectively exploiting the intermittently available idle computing, storage and communication resources in a Grid infrastructure. Specifically, we develop efficient resource discovery policies that can ensure continuous access to information sources and maintain application Quality-of-Service (QoS) requirements, e.g. required network transmission bandwidth on the mobile clients. Our performance studies indicate that mobility patterns obtained via tracking or user-supplied itineraries assist in optimizing resource allocation. The proposed policies are also resilient to dynamic changes in the availability of grid resources.

1 Introduction

The emerging digital wireless network infrastructure is being fueled by a number of factors such as the development of small and powerful mobile devices and the rising use of such mobile devices for a variety of business and entertainment applications.Mobile multimedia applications still present a significant challenge due to (a) large storage, bandwidth and computation requirements and (b) limited memory, computation and power on handheld devices. Furthermore, the mobile nature of the clients leads to frequent tearing down and reestablishment of network connections; the latency introduced by this process causes jitters which may be unacceptable for delay-sensitive streaming multimedia applications. Solutions such as *Mobile IP* introduce additional latency. While buffering at the client is a usual solution to jitter problems, mobile hosts often have limited memory resources.

One possible solution to achieve real time delivery guarantees in mobile environments is to use connection-oriented services combined with stream buffering in the path of service [18]. Another popular technique is to transcode [3] the incoming stream from the server at a local proxy that can customize the MM content based on user characteristics. Since different users require information at different QoS levels and devices may have varying resource/power capabilities, personalized customization of applications can achieve QoS assurance while prolonging the lifetime of the mobile device. Our objective is to use locally available (idle) grid resources to customize multimedia applications for mobile users based on user requirements and device limitations.

Several complications arise in ensuring the effective utilization of grid resources for mobile multimedia services. Firstly since grid resources are intermittently available; optimal scheduling policies must take the availability of grid resources into account. Secondly, the heterogeneity of grid resources and clients [12] complicates the resource management issue. Thirdly, user mobility patterns may not be known. Furthermore, since MM applications have QoS requirements (e.g. required network transmission bandwidth, accuracy and resolution of displayed images); an effective resource allocation policy must address the performance-quality tradeoff. This tradeoff arises since finding optimal local resources for a mobile host that lowers overall network traffic (i.e. improves performance) can introduce frequent switches in the multimedia stream possibly leading to increased jitter (lower QoS).

The rest of this paper is organized as follows. Section 2 describes a grid-based mobile environment for a video streaming service, and introduces the middleware architecture for such an environment. Section 3 introduces GRAMS (Grid Resource Allocation for Mobile Services), a generalized resource discovery algorithm and propose a family of GRAMS policies to address the dynamic nature of the mobile grid environment. We evaluate the performance of the GRAMS policies under different resource and mobility conditions in Section 4, and conclude with related work and future research directions in Section 5.

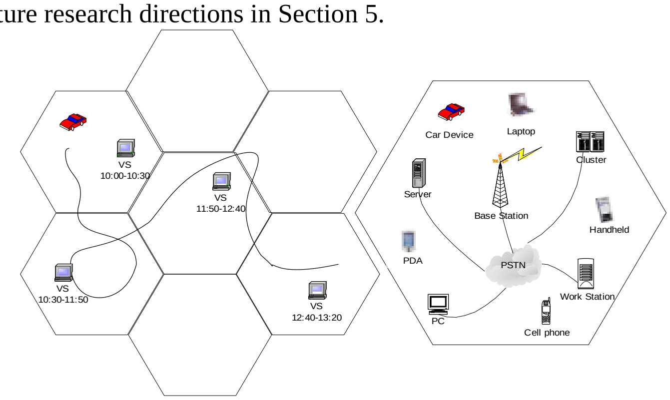

Fig. 1. System environment: (a) An example of providing service for a mobile client by Volunteer Servers available in grid system, (b) Possible components within a cell

2 A Middleware Framework

Figure 1(a) depicts the system environment in which a mobile user submits a request for streaming video data and traverses through a series of "cells" before arriving at a final destination[1]. Figure 1(b) illustrates the possible components within a cell that

[1] We consider mobile clients in infrastructure-based wireless, also we do not address the issues of wireless channel allocation (typically at MAC layer).

might be involved in the MM session such as: base stations, *Volunteer Servers* (*VS*s) and mobile clients. When not being used, a *Volunteer Server* (*VS*) participates in the grid by supplying idle resources, i.e. *VS*s are intermittently available. A *VS* can be a PC, work station, server, cluster, etc, which can provide high capacity storage devices to store the multimedia data, CPU processor for decompression and/or transcoding, buffer memory, and NIC (network interface card) resources for real-time multimedia retrieval and transmission. The *VS*s studied in this paper are fixed wired machines, whereas the mobile hosts connect to the infrastructure using a locally available wireless network. The mobile clients can be a PDA, handheld, laptop, or any wireless devices that can download and play video. They move around the cells, communicate with the base station, and hand in requests for the multimedia services.

Our approach is to divide the whole service period into non-overlapping chunks (possibly of different sizes). Subsequently, we attempt to map each chunk to an appropriate *VS*, for example one that is geographically close and lightly loaded. Video objects are also divided into equal sized segments and corresponding video segments are downloaded on the selected *VS*. The *VS* processes the request by transcoding the video segments and transmits the video stream to bandwidth limited and performance limited mobile clients, such as PDAs via wireless links. For example, in Fig 1(a), a mobile user goes through 5 cells. The service may be divided into 4 chunks that are served by various *VS*s. By choosing a *VS* close to the mobile client for performing the adaptation, the network transmission delay and overhead is reduced. Note that this approach has the additional benefit of accommodating varying wireless bandwidth in the local region. Video segmentation [4, 10], more intelligent caching [6], synchronization with handoffs [14] and high speed wireless transmission [21] make our approach feasible. In this paper, we assume that play-out buffer is correctly used to mask jitters by VS switches.

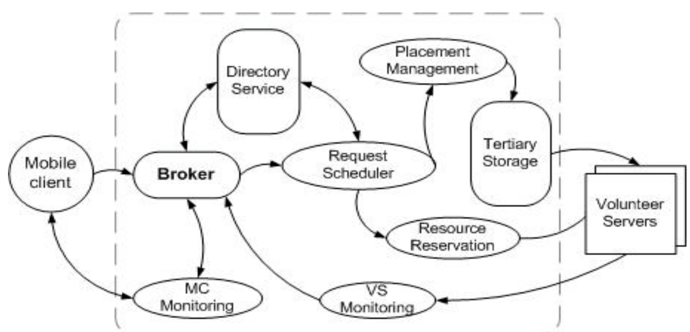

Fig. 2. A middleware service architecture

Fig 2 illustrates the various middleware components for discovering intermittently available resources for mobile multimedia applications. The key components are the *Broker* that performs admission control for incoming requests and manages their rescheduling; and the *Directory Service* module that stores information about resources availabilities at the *VS*s and clients. When a mobile client (*MC*) hands in a request, the *Broker* sends the request to the *Request Scheduler* which executes the resource discovery algorithm based on the resource and network information retrieved from *Directory Service*. If no scheduling solution exists the *Broker* rejects the request;

otherwise, the *Resource Reservation* module reserves the resource and the *Placement Management* replicates video segments from *Tertiary Storage* on each selected *VS* accordingly. The *Directory Service* is then updated with the new allocation information. If *VS* availability is changed, multiple preassigned requests may be invalidated. The *VS Monitoring* module detects dynamic changes in *VS* availabilities; the *Broker* is informed of these changes and the new configuration is registered in the *Directory Service*. Requests preassigned to *VS*s that fail unpredictably will need to be rescheduled, but if such requests cannot be rescheduled they will experience completion failures. The *MC Monitoring* module keeps track of the client's moving patterns. Changes detected by monitors can be reported to the *Broker* so that the necessary rescheduling processes will be triggered.

3 Grid Resource Allocations for Mobile Services (GRAMS)

Our prior work [17] uses a graph-theoretic approach to address the resource scheduling problem in multimedia grid environments for static clients. We now propose the GRAMS (Grid Resource Allocation for Mobile Services) algorithm to find suitable *VS*s for mobile hosts in a grid system. The objectives of the GRAMS process is to (a) execute load-based adaptive *VS* selection; (b) satisfy user QoS requirements; (c) bound the number of *VS* switches during the lifetime of a request; and (d) adapt to dynamic changes in user mobility and *VS* availabilities.

3.1 Notations and Definitions

This section defines the terms that will be used in the rest of this paper.

LoadFactor of a *VS*: The amount of available resources on a *VS* can vary over time, so we divide the entire duration of a day into time units, e.g. 10-minute time units. For each time unit, we consider four resources on a *VS* that must be allocated for every incoming request - CPU, memory (MEM), network bandwidth (NBW) and disk bandwidth (DBW). In order to deal with the capacity of each *VS* over time in a unified way and represent how much a request will affect the server during the requested period of time, we define a *LoadFactor*, LF_t for a request ***R*** on ***VS j*** at a particular time unit ***t***, as:

$$LF_t = \text{Max}\,[CPU_a, MEM_a, NBW_a, DBW_a] \text{ where: } X_a = R_X / S_{AvailX}(t);\ R_X \text{ is the requested resource X and } S_{AvailX}(t) \text{ is the amount of resource X available at time t.} \quad (\mathbf{1})$$

Thus, the ***LoadFactor*** LF_t is determined by the bottleneck resource during time unit ***t***. However, the duration of one requested period ***T*** may cover multiple time units; we therefore use the average *LoadFactor* over the time units.

Focus of the request for a given time period: represents a central point in space where a *MC* is likely to spend the maximum amount of time during a given chunk of a request. When without knowing the mobility pattern, we use the starting position as the focus, and adjust it at runtime using information from mobility tracking. Note that the entire request can be divided into chunks with each chunk having its own focus. Section 3.3 presents an algorithm for calculating the focus.

VS factor: In order to evaluate the benefit of choosing VS_j for a time period T of request R, we define a *VS factor* based on VS_j availability, the current load and the distance between the VS_j and the focus of the request R as:

$$VS\ factor(j,R,T) = \frac{ava(j,T)}{load(j,R) * dist(j,R)}, \quad where: load(j,R) = LoadFactor(j,R,T); \tag{2}$$

$$ava(j,T) = \begin{cases} 1, & if\ VS_j\ is\ available\ at\ T \\ 0, & otherwise \end{cases}; \quad dist(j,R) = dist\ from VS_j\ to\ R's\ focus during T.$$

MM Segment: A multimedia object with total service duration R_d (if continuous execution) is divided into N uniform sized segments of size S_d. ($R_d = N^* S_d$).

3.2 A Generalized GRAMS Solution

Given a request $R(VID, itinerary)$, where *VID* identifies the video object requested by the *MC* and the *itinerary* contains mobility information of the host (NULL, if no mobility information is available), we determine an appropriate *VS* allocation. A generalized GRAMS algorithm (See Fig 3) has three main steps. (1) *PartitionServicePeriod()* partitions the whole request service period into a fixed number of chunks that can potentially determine the number of *VSs* to be used for the request. (2) *VolunteerServerAllocation()* maps the chunks to specific *VSs* based on the load, availability and proximity. (3) *MobilityBasedRescheduling*() keep track of the *MC*. If the *MC* moves far away from the preassigned *VS*, then rescheduling may be triggered. We will describe each step in the following sections.

A Generalized GRAMS Algorithm
```
BOOLEAN found = true;
PartitionServicePeriod(); // step 1
FOR each chunk
        IF (VolunteerServerAllocation() == null)  THEN found = false;
IF (found) THEN MobilityBasedRescheduling() ELSE Reject the request;
```

Fig. 3.. generalized GRAMS

Step 1. Partition Service Period: We initially partition the service period using one of two possible strategies: (a) divide the whole service into uniform sized chunks, in multiple of S_d, (b) use a *Faststartup* Partition, which attempts to reduce the replication latency for the first segment of the video replica by choosing the first chunk as short as possible, and treats the remaining service time to be one big chunk.

Step 2. Volunteer Server Allocation: We choose multiple *VS*s to startup the continuous media service immediately; this is implemented by a network flow algorithm devised in [17]. Specifically, for each chunk of the service period, we choose the *VS* with the largest value of *VS factor*. If it is possible to find *VSs* for all chunks, there exists a scheduling solution; otherwise, the request is rejected. To calculate the *VS factor* (defined in section 3.1), we use the Euclidean distance between the *VS* and the chunk focus. Without knowledge of mobility pattern, we use the starting point to be the focus, and we present optimizations to the *VS* selection process in section 3.3 when knowledge of the host itineraries is available.

Step 3. Mobility-based Rescheduling: generally, randomness of client mobility will not affect the service completion ratio once the request has been scheduled, as the *VS*s connect to *MC*'s access point (base station) via a wired network. However, the pre-assigned *VS* may no longer be optimal due to changes in client itinerary; increased path lengths can introduce additional delays, jitter and network traffic. The system architecture can be enhanced by adding a mobility tracking module, which keeps track of *MC*'s location, and informs the broker to trigger rescheduling. Given the paper's limited space and the module's complexity, this issue is not addressed here.

3.3 Optimizations to GRAMS

In this section, we propose optimizations to step 1 and step 2 of the generalized GRAMS algorithm given knowledge of client mobility patterns.

Optimization of step 1 (Partition Service Period): The itinerary is specified as a series of times when a host moves into a new cell, e.g. $[(Time_1, Cell_1), ...(Time_N, Cell_N)]$. We first count the duration the *MC* stays in each cell, then partition the whole service period into chunks using two polices. Policy 1: Temporally biased, *MajoritySpreadover* Partition, which attempts to minimize the number of *VS* switches. We first choose the majority interval (cell with longest D_i) as the centerpoint of the service time; then spread the service time forward until the request submission time. If the entire service duration is not satisfied, we spread beyond the centerpoint until service can be completed. Note this policy may bring a period of startup latency between the submission and the startup time. Policy 2: Spatially biased, *Distance* Partition, which partitions the whole service time into chunks based on the distances between the cells traversed. We apply a well known unsupervised neural learning technique, self-organizing map (SOM) [19] to partition the whole moving area into a number of regions; therefore, the service period will be partitioned into chunks accordingly.

Optimization of step 2 (Volunteer Server Allocation): Given *MC*'s itinerary, we can determine the chunk focus in step 2 more accurately. If (a_i, b_i) represents the coordinates of the center for cell i, and D_i represents the time duration spent in cell i, we convert the problem of locating the chunk Focus into a Minisum Planar Euclidean Location Problem [15]. Our objective is to minimize the overall service time $f(x, y)$, where (x, y) represent the coordinates of the Focus position for this chunk that is composed of N cells. We calculate $f(x, y)$ by applying Weiszfeld's algorithm. [15]

$$f(x, y) = \sum_{i=1}^{N} Di\sqrt{(x - a_i)^2 + (y - b_i)^2} \cdot \qquad \textbf{(3)}$$

3.4 Dealing with Dynamic Changes in VS availability

A schedulable request may not be completed eventually due to dynamic changes in *VS* availability. To achieve system robustness, we reallocate other *VS*s for the interrupted requests to fulfill the remaining service. When a specific VS becomes unavailable, the broker retrieves information about requests that are scheduled on the unavailable *VS*, and triggers the re-scheduling process for each invalidated service. If requests cannot be rescheduled, the broker reports a request failure, in which case any resources reserved for this failure request on other selected *VS*s for the remaining service should be released. Devising optimal online approaches to deal with dynamic changes in *VS* availability is beyond the scope of this paper.

4 Performance Evaluations

In this section, we analyze the performance of the resource discovery strategies under various *VS* configurations and host mobility patterns.

4.1 The Simulation Environment

We simulate a cellular network system with 100 cells, with 50 *VS*s distributed evenly. Each *VS* has a storage of 100 GB and network bandwidth of 100Mbps. For simplicity, CPU and memory resources of the *VSs* are assumed not to be bottlenecks. We set the duration of each video that ranges from 1 to 3 hours; each video replica requires 2 GB disk storage, and network transmission bandwidth that ranges from 500 kbps to 2 Mbps. The shortest segment of a video object can run for 10 minutes. These parameters can be varied to simulate different configurations.

Request Model: Incoming multimedia requests are characterized by using a Zipfian distribution [9]. The request arrivals per day for each video V_i is given by:

$$\text{Pr. }(V_i \text{ is requested}) = \frac{K_M}{i}, \text{ where } K_M = \left(\sum_{i=1}^{M} \frac{1}{i} \right)^{-1}. \quad \textbf{(4)}$$

The probability of request arrivals in an hour j will be:

$$p_j = c / j^{1-f}, \; c = 1/(\sum (1/j^{1-f})) \text{ for } 1 \le j \le 24; \quad \textbf{(5)}$$

where Φ is the degree of skew and is assumed to be 0.8.

Mobility Patterns: We use the incremental mobility model [13] to characterize random mobility of each MC, and study three main host mobility patterns: (a) *Random movement*: the *MC* will travel at any speed, heading in any direction. (b) *Constant velocity with intermediate halt:* a linear and straight moving pattern with one long stop where the *MC* remains stationary. (c) *Clustered movement*: the *MC's* moving area can be geographically partitioned into at least 2 clusters.

Time map: We use the service time map to keep the information of when the *VS* will be available during the span of a day. We apply various approaches in modeling the time map. However due to the space limitation, we only discuss the model of *Uniform availability*, whose results are representative. *Uniform availability* – All *VS*s are available (or unavailable) for an equal amount of time; and the *VS*s are divided into groups such that within each group the time distribution covers the entire 24-hour day. In our simulations, we use the uniform availability with duration set to 6 hours as the basic time map strategy. We also executed the entire set of experiments with a variety of continuity or total availability for time maps.

4.2 Experimental Results

We evaluate the performance of various policies by using the number of rejections over execution time as the main metric. The average distance from the selected *VS* and request and the service completion ratio are also used in evaluations.

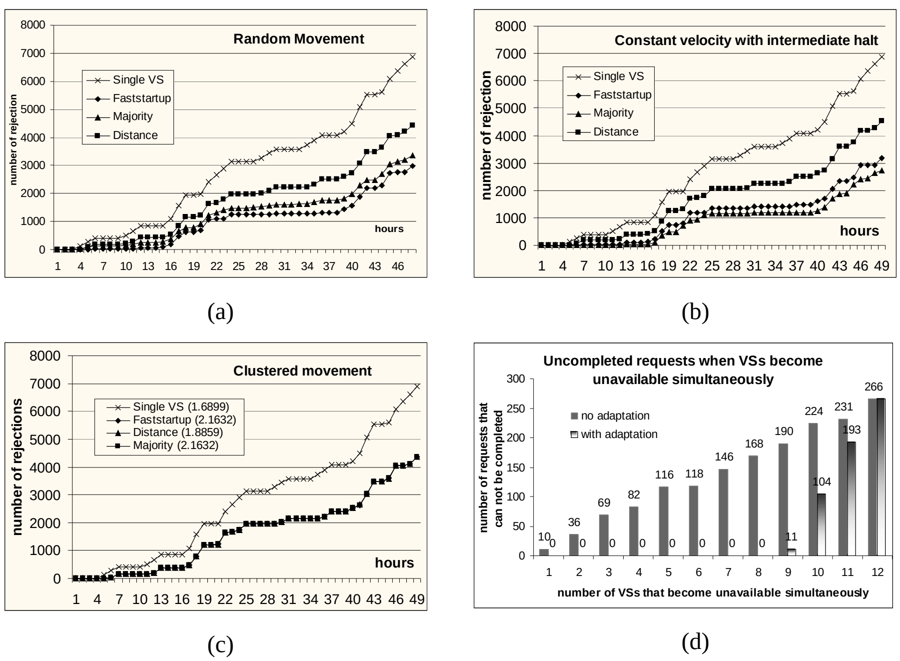

Fig. 4. (a), (b) and (c) illustrate experiment results for various policies under different moving patterns; (d) shows experiment results under dynamic changes in VS availability

4.2.1 A Comparison of GRAMS scheduling policies

We discuss the performance of four policies under different request models (Fig 4 a,b,c). The time map of each VS in Fig 4 (a,b,c) follows a ***Uniform availability*** pattern. The *Single VS* views the itinerary as a whole chunk with the service time R_d. Two policies (*Single VS* and *Faststartup*) assume no knowledge of mobility patterns, while the remaining two policies (*MajoritySpreadover* and *Distance* Partitioning) exploit the knowledge of user itineraries. When using the *Random* mobility model, the *Faststartup* policy performs the best with the least number of rejections (Fig 4a). However under the *Constant velocity with intermediate halt* model, the *MajoritySpreadover* policy has the least number of rejections (Fig 4b). Finally, when using *Clustered movement*, three of the policies (except *Single VS*) have similar rejection ratios, but *Distance* partition performs better by selecting *VS*s closer to the request (In Fig 4c, the number in the parenthesis marks for each policy represents the average distance from selected *VS* to the request). In other words, the results match the intuition behind the development of these policies. As can be observed, knowledge of the mobility model can help improve GRAMS performance significantly, note that knowledge of user itineraries can help avoid unnecessary advance replications to *VS*s that may be unused due to mobility-based rescheduling.

4.2.2 Dynamic Changes in *VS* Availability

When dynamic changes to VS availability occur, a number of factors affect the overall request completion ratios - the initial time maps of each *VS*, and the current load. Given the limited space, we only present the experiment results under a *Uniform availability* time map, *Random* mobility pattern, and a *Faststartup* policy. During the simulation, we have a total of 12 *VS*s available at the moment when changes happen. We gradually increase the number of simultaneous *VS* failures and observe its impact on request completion ratios. As Fig 4d 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 3.4. When the number of simultaneous VS failures increases to the extent that a large portion of the grid is unavailable (more than 8 *VS*s in our case), there are fewer overall resources available, causing increasing numbers of request completion failures.

We also study the impact of various time map models, cell sizes and host velocities on different policies. In summary, the GRAMS approach and adaptations are resilient to dynamic changes in *VS* availabilities and can significantly improve request completion ratios under unpredictable system conditions.

5 Related Work and Concluding Remarks

Streaming multimedia information to mobile devices has been addressed in [1, 16, 20]. Resource management in the grid has been addressed by several metacomputing projects such as Legion [8], Globus [11], AppLes [7], GrADS [5] and Nimrod [2], etc. Support for MM applications in the grid environment has been addressed in the context of Globus via the definition of a QoS component called Qualis, where low level QoS mechanisms can be integrated and tested.

In this paper, we have proposed service partitioning algorithms for mobile environments that effectively utilize available grid resources to significantly improve system performance. We show how to take advantage of apriori knowledge of mobility patterns to tailor the scheduling policies for better overall performance and enhanced user QoS. Resource discovery mechanisms for mobile grid environments must be resilient to changes in user mobility patterns and server time maps. We have developed effective extensions to the GRAMS policies to deal with the dynamic changes in VS availability and mobile host itineraries. Future work will address the degree of location awareness required by the middleware for efficient allocation of grid resources in a scalable fashion. A specific extension is to allow for a distributed brokerage service that will allow for localized resource discovery. We are also looking into tradeoffs that arise when timeliness requirements interfere with other application requirements such as security and reliability.

References

1. P. Agrawal, S. Chen, P. Ramanathan and K. Sivalingam, *Battery Power Sensitive Video Processing in Wireless Networks*, Proceedings IEEE PIMRC'98, 1998.
2. D. Abramson, J. Giddy, and L. Kotler, *High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid?* International Parallel and Distributed Processing Symposium, 2000.
3. P. Shenoy and P. Radkov, *Proxy-assisted Power-friendly Streaming to Mobile Devices*, In Proceedings of the Multimedia Computing and Networking (MMCN) Conference, 2003.
4. S. Chen, B. Shen, S. Wee, and X. Zhang, *Adaptive and lazy segmentation based proxy caching for streaming media delivery,* Proc. of ACM International Workshop on Network and Operating Systems Support for Design Audio and Video, 2003.
5. F. Berman, A. Chien, K. Cooper, J. Dongarra, I. Foster, D. Gannon, L. Johnsson, et. al. *The GrADS Project: Software Support for High-Level Grid Application Development,* International Journal of High Performance Computing Applications, 2001 pp. 327-344.
6. W. H. O. Lau, M. Kumar and S. Venkatesh, *A cooperative cache architecture in support of caching multimedia objects in MANETs,* Proc. ACM international workshop on Wireless mobile multimedia, 2002.
7. F. Berman and R. Wolski. *The AppLeS project: A status report*. Proceedings of the 8th NEC Research Symposium, 1997.
8. S.J.Chapin, D. Katramatos, J.F. Karpovich, A. Grimshaw, Resource Management in Legion, University of Virginia Technical Report CS-98-09, 1998.
9. A. Dan and D.Sitaram. *An online video placement policy based on bandwidth to space ration (bsr)*. SIGMOD, 376-385, 1995.
10. A. Dan, M Kienzle, D Sitaram, *Dynamic Policy of Segment replication for load-balancing in video-on-demand servers.* ACM Multimedia Systems, 1995.
11. I. Foster, J. Geisler, W. Nickless, W. Smith, S. Tuecke. *Software Infrastructure for the I-WAY High Performance Distributed Computing Experiment*. Proc. IEEE Symposium on High Performance Distributed Computing1997
12. D. Xu, D. Wichadakul, and K. Nahrstedt., *Multimedia Service Configuration and Reservation in Heterogeneous Environments*. International Conference on Distributed Computing Systems, 2000.
13. Z. Haas. *A new routing protocol for the reconfigurable wireless networks,* In Proceedings of the IEEE Int. Conf. on Universal Personal Communications, 1997.
14. C. K. Hess, D. Raila, R. H. Campbell, D. Mickunas. *Design and Performance of MPEG Video Streaming to Palmtop Computers*. Multimedia Computing and Networking 2000.
15. Philip M. Kaminsky, IEOR 251, Logistics Modeling. http://www.ieor.berkeley.edu/~kaminsky/ieor251/notes/2-3-03b.pdf
16. A. Joshi. *On proxy agents, mobility and web access*. ACM Journal of Mobile Networks and Applications, 2000.
17. Y. Huang, N. Venkatasubramanian, *QoS-based Resource Discovery in Interminttently Available Environments*, 11th IEEE High Performance and Distributed Computing, 2002.
18. R. H. Katz, *Adaptation and Mobility in Wireless Information Systems*, IEEE Personal Communications, 1994.

19. T. Kohonen, K. Mäkisara, and T. Saramäki, *Phonotopic maps - insightful representation of phonological features for speech recognition*. Proc. International Conference on Pattern Recognition, 1984
20. K. Keeton, B. Mah, S. Seshan, R.H. Katz, D. Ferrari, *Providing Connection-Oriented Network Services to Mobile Hosts*, Proceedings of USENIX Symposium on Mobile and Location-Independent Computing, 1993.
21. *3rd Generation Mobile Wireless*, A presentation on the Opportunities and Challenges of Delivering Advanced Mobile Communications Services, 2002.