Optimizing Operation of a Hierarchical Campus-wide Mobile Grid for Intermittent Wireless Connectivity

Konstantinos Katsaros and George C. Polyzos
Mobile Multimedia Laboratory
Department of Computer Science
Athens University of Economics and Business
Athens 104 34, Greece
ntinos@aueb.gr, polyzos@aueb.gr

Abstract— **Recent advances in mobile communications and computing and strong interest of the scientific community in the *Grid* have led to research into the *Mobile Grid*. We discuss various approaches proposed in the literature and try to point out the fundamental issues and problems emerging from the introduction of mobile devices and wireless communications in the context of the Grid computing paradigm. We further propose an architecture for the realization of a Mobile Grid and investigate key design decisions and optimizations.**

I. Introduction

Grid computing has emerged as a paradigm for the coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations [1]. A grid computing system is essentially a large-scaled distributed system designed to aggregate resources from multiple sites. Users of such systems have the opportunity to take advantage of enormous computational, storage or bandwidth resources that would otherwise be impossible to attain. In many cases these resources would be wasted if not aggregated inside a grid.

On the other hand, the relatively recent advantages in mobile and wireless communications have resulted in the availability of an enormous number of mobile computing devices such as laptop PCs and PDAs. In effect, it is considered natural to extend the idea of resource sharing to mobile and wireless communication environments. However, there are various, quite different approaches on the exact character of this extension [2], [3], [4], [5], [6], [7], [8], [9], [10], [11]. Their differences mostly rely on whether mobile devices are considered powerful enough to provide their resources or not. In agreement with [3], we believe that since the number of available mobile devices is nowadays enormous and their computational power is constantly increasing, the aggregated sum of their resources should be exploited . Especially in environments such as university campuses, the number of available mobile devices is in the order of thousands (e.g. almost 6200 distinct MAC addresses recorded in the traces of Kotz *et al.* [12]). Considering the fact that such networking environments are usually under a single network administration, we believe that accumulating these concentrated and potentially unexploited resources is an interesting and significant challenge.

In this paper, we propose a campus-wide hierarchical Mobile Grid system architecture in which mobile nodes (MNs), willing to offer their computational resources, move between WLANs. This willingness is based on *reciprocity*. We consider *divisible load* [13] applications (e.g. volume rendering) where the load of computation can be divided into several parts i.e. a job that can be divided into tasks that can be carried out independently of each other. A list of such applications is presented in [14]. In order to investigate the feasibility and gain insights on the performance of the proposed architecture we utilized WLAN traces collected at Dartmouth College [12]. These traces provided a realistic simulation environment suitable for the study of the performance of a hierarchical campus-wide Mobile Grid system.

The remainder of the paper is organized as follows. In Section II we examine related work. In section III we describe the proposed architecture and in section IV we present the framework for the evaluation of the proposed system. In section V we present the results of the performance evaluation. We finally conlude in section VI and present our plans for future work.

II. Related Work

A. Mobile Devices as Resource Consumers

In this case, mobile devices are considered to have limited computational and/or storage capabilities [4], [7], [9], [10]. Of course this is true for mobile devices in general and actually this is the main reason for which the supporters of this first approach propose the integration of mobile devices to the grid. The grid can provide the resources missing in mobile devices on demand. Nevertheless, problems related to the very nature of wireless and mobile devices arise here. Frequent disconnections and limited battery life make it difficult to directly interact with a grid system. Submitting jobs and receiving the results back is not as straightforward as it may seem, since power constraints and frequent disconnections are prevalent in wireless and mobile communications. The use of proxies is proposed in [4] which act as gateways to the grid. These proxies undertake the role of the mediator between the mobile device and the grid system, and try to hide the instability of the wireless/mobile environment by being responsible for submitting the job, monitoring its execution and returning back the results i.e. acting on behalf of the mobile device.

Other approaches on solving the resource limitation problem of mobile devices target at providing a ”smart” environment for *pervasive* computing [7], [9]. In these approaches, mobile devices are considered as pure access devices without need for enhanced processing and/or storage capabilities [10]. The role of the grid is to provide all the functionalities required by users pushing this way the complexity of the whole system to the networking environment rather than to the edges.

B. Mobile Devices as Resource Providers

In this case, strong emphasis is given to two important factors. First, even though mobile devices have limited resources compared to their stationary counterparts, they seem to increasingly gain sufficiently powerful CPUs and storage means. In effect, they are considered capable of providing useful resources, whenever these are not used by the user. Second, since the number of mobile devices continuously increases, the aggregate of their resources cannot be considered negligible [3]. However, again, mobility related problems such as limited power and intermittent connectivity pose significant difficulties in the effort of exploiting these resources. A mobile device, for example, may be capable of executing a specific task but it may not be able to return the results back in time due to an unexpected disconnection.

Two fundamentally different architectures have been followed in an effort to exploit resources relying in mobile devices: mobile grids on-site and mobile ad-hoc grids.

1) Mobile Grids On-Site: In the case of creating mobile grids on-site [3], [4], [5], [6], mobile devices residing in a well defined area such as a cell in cellular networks or a WLAN hot-spot (Service Area, SA) are coordinated by a central entity (residing at the Access Point/Base Station, BS) in order to perform a task (computation grid). In this approach, the devices provide the description of their capabilities and the degree of their availability to the BS. The BS is then responsible for decomposing an incoming request and scheduling the overall execution by providing specific tasks to each of the participating mobile devices. A request can either come from a mobile device in the SA or another client outside this area (fixed or mobile). The advantage of this approach is that the BS can act as a mediator capable of hiding the heterogeneity of the participating devices from the requesting node, coordinating the overall execution of the submitted job and even allow the grid system to appear to the rest of the network as an ordinary grid node [3].

In the architectures presented in [5] and [6] the authors do not actually address the problem of mobility. If a mobile node leaves the SA the task that it was executing is aborted in the sense that it will be rescheduled. In effect, the proposed schemes result in the waste of resources for this mobile node. Moreover, the task may complete successfully while the mobile node is located in the neighboring SA and the results could be returned from that point of attachment to the network. On the contrary, as discribed in Section III, our architecture considers this issue by allowing MNs return their results possible from different points of attachment to the network. It is obvious that there is a *distinction between disconnection and failure* [7].

2) Mobile ad-hoc grids: In the case of mobile ad-hoc grids [8], [15], there is no central authority responsible for the coordination of the overall job execution. Further problems arise in this case due to the ad-hoc nature of such systems. The absence of central coordination imposes difficulties in service discovery, job scheduling and monitoring. An approach towards overcoming this limitation, is the formation of a virtual backbone consisting of a number of, possibly more powerful, mobile nodes responsible of coordinating the mobile nodes residing in a certain area of the overall ad-hoc network[8], [15]. The instability of the network topology induces further difficulties due to unique ad-hoc related characteristics such as network partitioning and multi-hop routing.

C. The Akogrimo Integrated project case

The EU IST project Akogrimo ”Access to Knowledge through the Grid in a Mobile World” [11] aims at the provision of a framework for deploying mobile grids focused on solving complex problems by exploiting knowledge as their main input. The distinction between this type of mobile grid and the ones described above is that the resource shared in this grid is *information*. In Akogrimo, data and computational power are accessed by dedicated fixed-grid providers [16].

III. Proposed Architecture

In this paper, we propose an architecture for a hierarchical campus-wide Mobile Grid. In this architecture we follow the Mobile Grid On-Site approach as described in II-B.1. In a typical university campus, multiple WLANs are deployed in each building. Students, faculty and administrative personnel move with their portable devices between WLANs and buildings. As shown in Figure 1, at each level of the hierarchy a Mobile Grid Scheduler (MGS) is responsible for receiving jobs from the upper layer MGS, splitting them into separate tasks and assigning these tasks to the lower layer MGSs. Each MGS is also responsible for communicating the available resources up to the hierarchy of MGSs, in order for upper-level MGSs to be able to know how to divide a submitted job i.e. enforce *load balancing* techniques. At the root of this tree-like structure the root MGS (R-MGS) can receive jobs from outside the campus i.e. clients not attached to the campus network. In a three level instance of this architecture, an intermediate MGS (I-MGS) located at each building can receive a task form the R-MGS and assign the produced sub-tasks to the local schedulers located at each WLAN in the building (L-MGS). Each L-MGS further splits the received task and assigns the resulting sub-tasks to the MNs residing at the WLAN it serves[1]. Participating MNs process the received

[1]It is noted that L-MGSs are not physically identified with APs because of the typically limited resources of the latter.

input data and return the results back to the assigning L-MGS. The transfer of the task input data, as well as the resulting output data, may be completed through one or more different WLANs (other than the initial one) due to the MNs' mobility. A MN may submit a job directly to the L-MGS at its WLAN. The L-MGS may retain responsibility for the job splitting and scheduling or it may propagate the job to a higher level of the schedulers' topology. The criteria for this decision as well as the load balancing techniques that could be deployed are subject to future work.

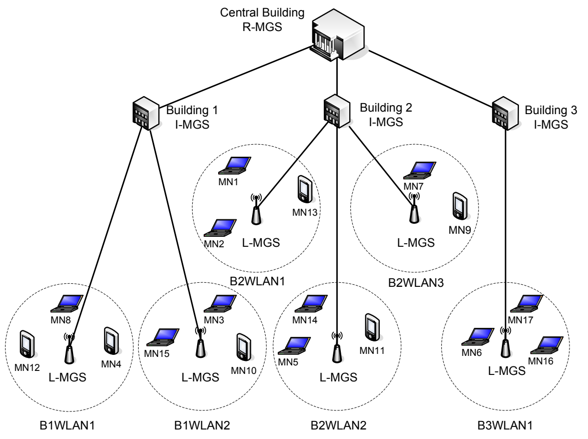

Fig. 1. Proposed architecture.

The hierarchical character of the proposed architecture presents significant advantages for the overall coordination of job decomposition and scheduling. A lower level scheduler hides all the details regarding the scheduling of a task it receives. I-MGSs need not be aware of each participating MN's networking behavior as this is a responsibility of L-MGSs. Essentially, this hierarchical approach on the design of the proposed architecture was followed to depict the *divide-and-conquer* nature of the job splitting, assignment and execution process. Furthermore, the distributed character of the proposed architecture ensures the operability of the overall system in cases of MGS failures. This also holds in the event of an R-MGS failure, since each L-MGS is capable of receiving job submissions directly from the MNs. In this case, only clients residing in networks outside the campus will not be able to submit their jobs. Moreover, a campus site is considered especially suitable for the deployment of the above described architecture because of its distributed structure, the existence of a central administration, which easies the control of the entire networking architecture, and the concentration of large number of mobile devices. For example, the WLAN traces collected at the campus of the University of Dartmouth [12] contain information about 6202 distinct wireless cards and 579 APs in 168 buildings. Although, it cannot be assumed that each wireless card corresponds to a single MN, this vast number indicates the not impermanent concentration of a large number of mobile devices, and hence their computational resources, in a well specified area.

A. Incentives

A critical issue in the deployment of the proposed architecture is the incentives given to mobile nodes in order to motivate them to offer their resources. As mentioned earlier, our scheme is based on *reciprocity*. A MN can submit a job to the Mobile Grid only if it shares its own resources as well. However, several issues must be addressed in order to ensure the viability and usefulness of the proposed system. The amount of resources offered by a MN to the Mobile Grid over a certain period of time, may not exceed the amount of resources required for the execution of its own jobs during the same time-period. Otherwise a MN will simply prefer not to participate in the Mobile Grid and compute its own jobs directly. Thus, the main target is to exploit the vast number of MNs by harvesting relatively small amounts of resources from each of them at a time and taking advantage of the parallel character of task execution. Suitable mechanisms must be designed in order to ensure the fairness of this scheme and furthermore to eliminate the danger of *free-riding*. An intelligent *accounting mechanism* is considered necessary in this effort.

Further investigation towards the design of a delicate incentives scheme together with an accounting mechanism that will ensure fairness is subject to imminent future work.

IV. Evaluation Framework

A. Formulation of the problem

The core functionality of an MGS in the proposed architecture is to receive a job and, in the context of *divisible load* applications [13], divide the submitted workload into tasks for submission to its descendant MGSs. At the lowest level, L-MGSs submit the produced sub-tasks to the MNs residing at the corresponding WLAN. The whole process consists of three distinct steps: the transfer of the input workload to the MN, the task execution and the return of the results back to the L-MGS. In the absence of disconnection events, each step requires T_{IN}, T_{EXEC}, T_{OUT} amount of time to complete, respectively. We also define T_{TOTAL} as:

$$T_{TOTAL} = T_{IN} + T_{EXEC} + T_{OUT}$$

The *communication to computation ratio* (CCR) [14] of a divisible load application is defined as :

$$CCR = \frac{CommunicationCost}{ComputationCost}$$

The $CommunicationCost$ factor denotes the time required by the MN to receive the task workload. The $ComputationCost$ denotes the time required by the MN to complete the execution of the submitted task.

B. Performance Constraints

The performance of the proposed architecture heavily depends on the *Response Time* (RT) of each MN working on a task. In ideal environments, where no disconnections take place, we obviously have $RT = T_{TOTAL}$ i.e. the MN receives the input data, completes their processing and returns the results to the L-MGS before getting disconnected. On the other hand, in a wireless and mobile context, intermittent connectivity introduces further delays on the completion of the whole process. The process of transffering the input/output data to/from the MN may be interrupted by a disconnection due to the MN's mobility and/or power constraints. Hence, a further delay on T_{IN} and/or T_{OUT} is imposed in this case. However, the very existence and the length of this delay directly depend on the length of the time periods spent by the MN in dis/connection. If a disconnection event occurs after a period of T_{TOTAL} time units, then we have no delays and RT equals T_{TOTAL}. It is important to note than disconnections do not affect the time required for the MN to process the input data since a MN may continue processing even if it is disconnected. This time is affected however by user actions, since mobile devices are strictly personal (e.g. a user may decide to turn of his device). In this case, an additional delay is imposed on T_{EXEC}. This delay is not investigated in the reminder of this paper.

The authors in [17] have concluded that, for the same collection of traces, the average length of a MN *session*[2] is 16.6 minutes, 71 % of sessions finish in less than an hour and 27% of the sessions last less than one minute. As these findings demonstrate the intermittent character of these MNs' connectivity, we investigated the extend to which the performance of the proposed Mobile Grid system is affected. Our findings are presented in the next section.

C. Task Replication

One important issue, affecting the performance of the overall system, is the division of the total workload into chunks for each MN in the WLAN, by the L-MGSs. As mentioned above, the higher the workload for the MN the longer the T_{TOTAL} and the higher the probability of facing disconnection delays which result in an even longer RT i.e. $RT > T_{TOTAL}$. Apparently, an intelligent approach is required in order to mitigate the overhead incurred by MNs' mobility.

We investigate the approach of *task replication* i.e. the assignment of a certain task to more than one of the MNs residing in the same WLAN. This approach is based on the fact that not all MNs present the same networking behavior. Therefore, if the same task is submitted to multiple MNs it is highly probable that one of them will eventually return the results earlier than the others. In this way, possible disconnections of the rest of the MNs are hidden from the scheduler. However, it must be noted that task replication unavoidably results in the waste of resources. If a certain MN returns the results of a computation task earlier than the MNs which have received the same task, then the resources of the remainder of the MNs are wasted. Moreover, careful design is required in order to avoid situations where an excessive number of MNs receives the same task. Apart from the apparent waste of resources, excessive task replication has another important side-effect. Since a job of a certain input, processing and output load is splited by the L-MGS in order to be completed in its entirety by the co-located MNs, the greater the extend of task replication the larger the size of each task will be. In effect, the probability of disconnection events also rises, as mentioned above.

In section V-B we present our findings on the proposed task replication mechanism.

[2]A *session* is defined as the period of time in which a user (card) joins the network, uses the network, possibly roams to other APs in the same subnet, and leaves the network.

D. Traces

In order to investigate the performance of the proposed architecture we utilized the WLAN traces available at [12]. These traces provided us with realistic information on the mobility and connectivity characteristics of each MN in the campus. For each unique wireless card the traces consist of the sequences of access points the card has associated with, accompanied by the corresponding timestamps. A special AP name ("OFF") is used when the card has disconnected from the wireless network. In order to easily retrieve useful information we inserted the traces into an RDBMS table. The traces where collected from April 2001 to March 2003. Due to holes in the traces during fall of 2001 we only used data collected during 2002 and 2003. These traces contain information for 5982 distinct wireless cards and 566 APs in 166 buildings.

V. Performance Evaluation

In order to study the performance of the proposed architecture we need to define suitable metrics. As discussed in the previous section, the RT of each MN is a suitable metric that indicates the time required for the MN to return the results back to the L-MGS. We chose to study the resulting RTs for various T_{IN}, CCR and T_{OUT} values. Instead of providing explicit values for the input and output loads (e.g. in MB) we chose to describe these factors in terms of *time* i.e. the time required for the transfer of input and output data in the absence of disconnection events. Our decision is based on the fact that we did not have any information on the actual throughput in the WLANs under study. An indication of the actual input and output loads may be derived from performance measurement studies. In [18], the measurements show a net throuput of 47% at 11Mbit/s, including the TCP/IP protocol stack overhead.

A. Delay Overhead

We first measured the overhead on the RT incurred by intermittent connectivity, for various values for the input load (i.e. T_{IN}) and the CCR. We chose to set $T_{IN} = T_{OUT}$, though a more detailed study of the input and output load parameters ratio is included in our plans for future work. The overhead is measured as follows:

$$Overhead = \frac{RT - T_{TOTAL}}{T_{TOTAL}} * 100\%$$

For clarity reasons we split the results into two separate Figures. The measured overhead for high CCR values is presented in Figure 2 and the measured overhead for low CCR values is presented in Figure 3.

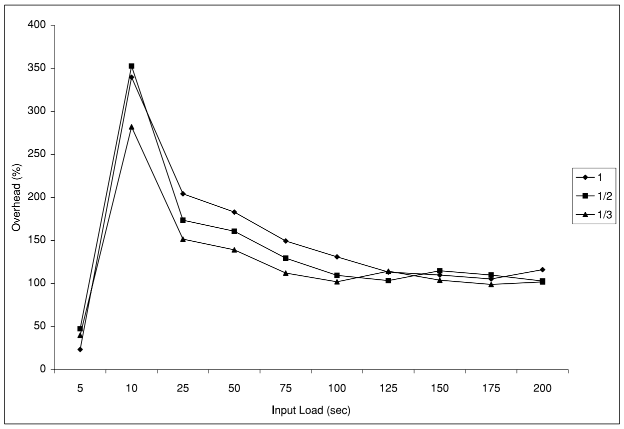

Fig. 2. Overhead for high CCR values

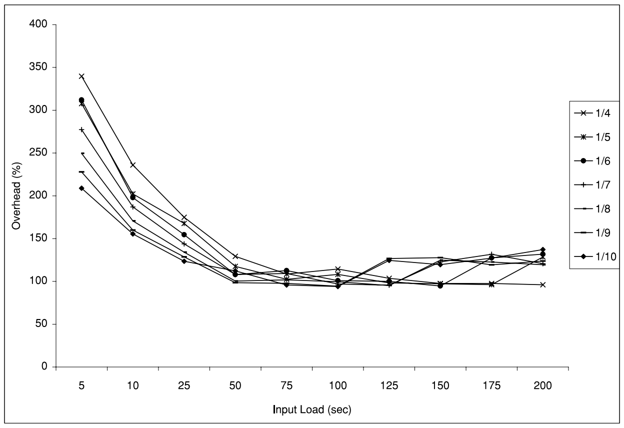

Fig. 3. Overhead for low CCR values

In Figure 2 we can see that for low input load, mobility incurs comparatively low overhead but there is a dramatic increase for greater input volumes right afterwards. As the input volume increases the overhead decreases. Initially, the input load is so low that the results are returned back almost immediately. This is because of the combination of low input load and high CCR. In effect, the resulting T_{TOTAL} is low enough for the most MNs to return the results back without suffering any disconnection delays. The following dramatic increase is owing to the disconnection events taking place during the overall task assignment, execution and return of results. As the input load continues to increase, the resulting overhead decreases. This is of cource not to imply that no disconnection events take place but that the disconnection periods are utilized for the computation part of the overall job ie. T_{EXEC}.

In Figure 3, we can see that for lower CCR values and low input load the overhead ranges from approximately 210% to 340% and starts decreasing as the input load increases. One would expect the opposite behavior but this is explained by the fact that, for these parameter values, the total load (T_{TOTAL}) is low and in effect any disconnection events result in increased overhead. Due to the low CCR values, T_{TOTAL} is longer than in the case of high CCR values and therefore the probability of disconnection during the transmission of the results increases. In the same figure, we surprisingly notice that for low input volumes higher CCR values result in higher overhead. However, this can be explained by the fact that for lower CCR values, major part of T_{TOTAL} is spent for the computation of the task (T_{EXEC}). In effect, disconnection periods are compensated by the execution of the task. For low input loads, we have less disconnection events and therefore more easily counterbalanced by the task execution period. Nevertheless, as the input load increases, so does the probability of disconnection events. Hence, for high input volumes we see an increase in the overhead. Notice, that for heavy input loads, high CCR values present better performance since they result in lower T_{TOTAL}.

B. Task Replication

In section IV-C we discussed the issue of task replication in the context of the proposed architecture. In order to decide on the extend of task replication i.e. the number of distinct MNs receiving the same task, we measured the average number of MNs co-residing in the same WLAN. On average, we found out that approximately 4 MNs are served by a certain AP at any given time. Based on this observation, we decided to study the aforementioned approach for the case of two distinct groups of MNs receiving the same tasks. This means that, in each testing environment, the MNs are splited into two groups. If N is the number of MNs residing at the WLAN, then the overall job is divided into $\frac{N}{2}$ tasks.

In order to investigate the performance of the aforementioned approach, we extracted approximately 1000 *testing environments* from the utilized traces. Each testing environment corresponds to a certain time and AP in the traces and refers to the MNs residing in the specific AP at the chosen point in time. All testing environments were uniformly extracted from the whole period of traces and the enirety of the APs.

We examined the performance of the aforementioned approach (*GROUPS* case) and compared it with the simple scenario of no task replication (*NO GROUPS* case),for various input load and CCR values. The results are shown in 4 and 5.

In Figure 4, which shows the performance gain relatively to T_{TOTAL} which corresponds to the time required by a single MN to receive the input data, complete the computation of the task and return the results back to the L-MGS in the absence of disconnection events, we can see that the *GROUPS* case clearly outperforms the *NO GROUPS* case. Contrary to the *NO GROUPS* approach, which incures an average overhead of 57%, the *GROUPS* case reduces the required time for the completion of the overall job, by 28% on average, leveraging this way the parallel character of task executions, for all input values. However, by examining Figure 5, which shows the percentage of testing environments in which each case yielded better performance, we find out that in the vast majority of environments, the *NO GROUPS* case is preferable compared to the *GROUPS* case. This means that following the *NO GROUPS* case results in the MNs returning their task execution results earlier in most of the cases. We conclude that this is because of the increased task size in the *GROUPS* case (actually double the task size in the *NO*

GROUPS case), which rises the probability of suffering a disconnection event before the completion of the transmission of the results. Our conclusion is supported by the fact that as the total load increases so does the fraction of the cases in which the *GROUPS* case is preferable. With an increased task load the MNs in the *NO GROUPS* case also face a higher danger of disconnection events before the completion of the task. In retrospect, the presented results clearly demonstrate the superiority of the *GROUPS* case in the presence of disconnection events.

Further investigation of the proposed mechanism, in order to balance between RT gains and resource waste (e.g. bandwidth consumption), is subject to future work.

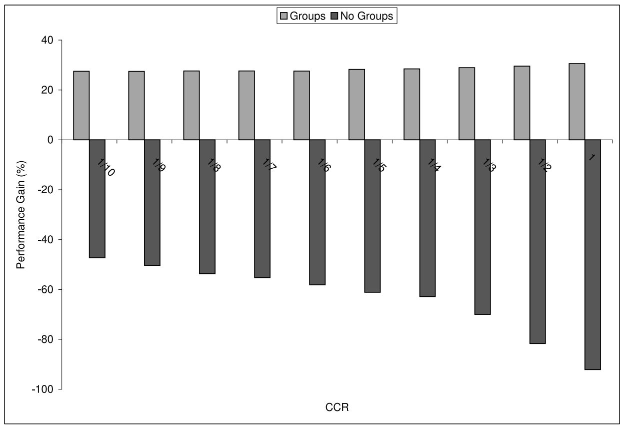

Fig. 4. Performance gains.

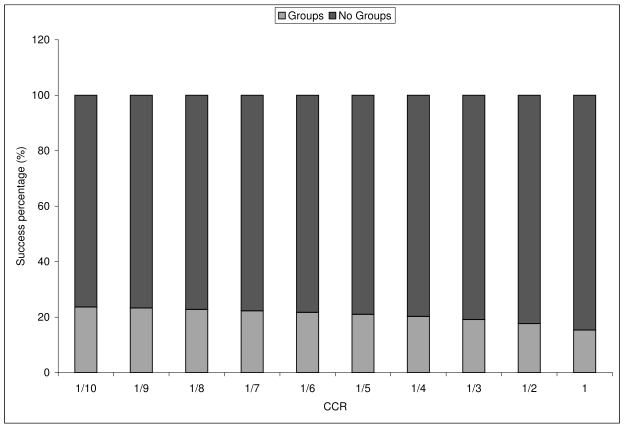

Fig. 5. Percentage of preferable situations.

VI. Future Work

In this paper, we investigated the fundamental issues rising in the path towards the realization of the Mobile Grid paradigm. We discussed various approaches in literature and pointed out the problems introduced by node mobility. Moreover, we proposed a realistic, hierarchical, campus-wide networking architecture and studied its performance based on real traces. We further proposed a simple task replication scheme for the mitigation of the intermittent connectivity problem.

Our plans for imminent future work include, but are not limited to, the investigation and detailed design of the incentives mechanism and the investigation of scheduling and load balancing techniques throughout the hierarchical structure of the proposed system. We further intent to investigate the effects on the performance of the system incured by the not negligible scenario in which MNs choose to abort the execution of a task.

References

[1] Ian T. Foster, "The anatomy of the grid: Enabling scalable virtual organizations," in *Euro-Par '01: Proceedings of the 7th International Euro-Par Conference Manchester on Parallel Processing*, London, UK, 2001, pp. 1–4, Springer-Verlag.

[2] A. Litke, D. Skoutas, and T. Varvarigou, "Mobile grid computing: Changes and challenges of resource management in a mobile grid environment," in *Proceedings of Practical Aspects of Knowledge Management(PAKM 2004)*, 2005.

[3] Thomas Phan, Lloyd Huang, and Chris Dulan, "Challenge:: integrating mobile wireless devices into the computational grid," in *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, New York, NY, USA, 2002, pp. 271–278, ACM Press.

[4] Sang-Min Park, Young-Bae Ko, and Jai-Hoon Kim, "Disconnected operation service in mobile grid computing.," in *ICSOC*, 2003, pp. 499–513.

[5] Stanislav Kurkovsky and Bhagyavati, "Wireless grid enables ubiquitous computing.," in *ISCA PDCS*, 2003, pp. 399–404.

[6] Stan Kurkovsky, Bhagyavati, and Arris Ray, "A collaborative problem-solving framework for mobile devices," in *ACM-SE 42: Proceedings of the 42nd annual Southeast regional conference*, New York, NY, USA, 2004, pp. 5–10, ACM Press.

[7] Guruduth Banavar, James Beck, Eugene Gluzberg, Jonathan Munson, Jeremy Sussman, and Deborra Zukowski, "Challenges: an application model for pervasive computing," in *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, New York, NY, USA, 2000, pp. 266–274, ACM Press.

[8] Lingfen Sun huoqun Li and Emmanuel C. Ifeachor, "Challenges of mobile ad-hoc grids and their applications in e-healthcare," in *2nd International Conference on Computational Intelligence in Medicine and Healthcare (CIMED2005)*, 2005.

[9] S. H. Srinivasan, "Pervasive wireless grid architecture.," in *WONS*, 2005, pp. 83–88.

[10] Mauro Migliardi, Muthucumaru Maheswaran, Balasubramaniam Maniymaran, Paul Card, and Farag Azzedin, "Mobile interfaces to computational, data, and service grid systems.," *Mobile Computing and Communications Review*, vol. 6, no. 4, pp. 71–73, 2002.

[11] "Akogrimo integrated project homepage," http://www.akogrimo.org.

[12] David Kotz, Tristan Henderson, and Ilya Abyzov, "CRAWDAD trace set dartmouth/campus/movement (v. 2005-03-08)," Downloaded from http://crawdad.cs.dartmouth.edu/dartmouth/campus/movement, Mar. 2005.

[13] Veeravalli Bharadwaj, Thomas G. Robertazzi, and Debasish Ghose, *Scheduling Divisible Loads in Parallel and Distributed Systems*, IEEE Computer Society Press, Los Alamitos, CA, USA, 1996.

[14] Cardinale Y. and Casanova H., "An evaluation of job scheduling strategies for divisible loads on grid platforms," in *High Performance Computing and Simulation Conference, 2006. (HPCS 2006)*, 2006.

[15] Marinescu C. Dan, Marinescu M. Gabriela, Yongchand Ji, and Boloni Ladislau, "Ad hoc grids: communication and computing in a power constrained environment," in *Performance, Computing, and Communications Conference, 2003. Conference Proceedings of the 2003 IEEE International*, 2003, pp. 113– 122.

[16] Martin Waldburger and Burkhard Stiller, "Toward the mobile grid: Service provisioning in a mobile dynamic virtual organization," in *4th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA-06)*, 2006, pp. 579–583.

[17] David Kotz and Kobby Essien, "Analysis of a campus-wide wireless network," in *Proceedings of the Eigth International Conference on Mobile Computing and Networking* (MobiCom'02). ACM, Sept. 2002.

[18] Kamerman A. and Aben G., "Throughput performance of wireless lans operating at 2.4 and 5 ghz," in *Personal, Indoor and Mobile Radio Communications, 2000. PIMRC 2000.*, 2000, pp. 190–195.