Modeling a Grid-Based Problem-Solving Environment for Mobile Devices

Stan Kurkovsky, Bhagyavati, Arris Ray
Department of Computer Science
Columbus State University
{Kurkovsky_Stan, Bhagyavati, Ray_Arris}@colstate.edu
Contact author: Stan Kurkovsky
Email: Kurkovsky_Stan@colstate.edu
Mailing address: Columbus State University
Department of Computer Science
4225 University Avenue
Columbus, GA 31907
USA

Abstract: *The paradigm of grid computing has been successfully applied in the domain of computationally-intensive applications supporting scientific research. Server-class computers interconnected by wired networks are typically used as computing devices in the grid. At the same time, the users of the existing and growing number of wireless mobile devices often demand more computational power than their devices can currently provide. We propose using the paradigm of the computational grid to build a problem-solving environment for wireless mobile devices. We address the challenges of distribution, coordination and assembly of a complex task, as well as such issues as network stability, access transparency and dependability in a grid-based system of mobile devices.*

Keywords: Grid computing, computational grid, wireless mobile device

Reviewed and accepted: 31 Mar. 2004

1. Introduction

Computational grids represent a new and rapidly evolving research area, which has gained a lot of attention in the past several years. A computational grid can be viewed as a transparent aggregation of many computing devices on a network that enables sharing of distributed resources. Typically, computational grids are considered in the context of sharing processor power of many computers interconnected by a wired network [1,3,4,17]. Most of the current grid implementations focus on high performance computing mostly supporting applications of scientific research. These include applications for massively parallel processing of raw data generated from processor-intensive simulations, such as the modeling of global weather patterns or large scale signal processing. Consequently, computing devices employed for such grid applications usually consist of homogeneous collections of resource-rich, server-class computers.

However, wireless mobile devices gained a tremendous popularity and their numbers continue to grow steadily. Such mobile devices are characterized by limited resources (slower processors, limited amount of memory, and short battery life) and by their high mobility. Given the wireless nature of their network connectivity, mobile devices also inherit many drawbacks of their transmission medium. The two major factors that cause wireless network instability are the nature of the transmission medium and the mobility of the devices [18]. Interference in wireless communications can be caused by atmospheric disturbances and other environmental factors such as terrain and vegetation. The other major cause of the network instability is the mobility of the devices involved in the wireless applications. Any wireless application involving simultaneous cooperation with other computing devices must be robust enough to shield the user from possible instability of the network by making network access transparent to the end user.

In addition to limited processing power, energy constraints imposed on mobile devices by small batteries present another motivation to move the computational load elsewhere [12]. Despite the small size and limited resources of mobile devices, their users demand more processing power. Typically, such demands are satisfied by advances in producing smaller and more powerful processors that consume less energy. As long as Moore's law holds, there is no reason to doubt that chipmakers and handheld device manufacturers will keep offering better and faster PDA's and smart phones. However, the paradigm of the computational grid can be applied to the domain of mobile devices to offer users more processing power using the existing hardware technology [13]. By redistributing the computational load on demand, a grid-based problem solving environment would enable the users of mobile devices to solve far more complex and resource-demanding problems that they would be able to solve using their individual devices on a stand-alone basis [6,15,16]. Using mobile devices in grid-based environments brings a lot of critical issues that need to be studied, primarily arising from the limitation of resources, wide heterogeneity of the population of existing mobile devices, and wireless network connectivity.

In this paper we present a prototype of a grid-based problem-solving environment for wireless mobile devices with limited processing power [2]. Its primary purpose is to allow mobile devices with limited resources to solve problems that they would not be able to solve individually. This goal is achieved by redistributing the computational load among many computing devices. The paper is organized as follows. Section 2 presents concept and architecture of the grid-based collaborative problem solving framework. Section 3 describes the details of design and implementation for the proposed environment. Section 4 discusses the issues of transparency and dependability in grid-based systems. Section 5 presents preliminary results and scenarios of simulation experiments that we will conduct to study the effects of device mobility and network stability on the behavior of the grid-based environment. Section 6 provides the summary and references to related literature are provided in Section 7.

2. Architecture of the grid-based problem solving environment

Previously, we proposed an architecture that embraces mobile devices (such as smart phones, PDAs, tablet PCs and wireless laptops) within a wireless cell to share their computational power by joining a framework modeled after a computational grid [2]. The architecture of the proposed architecture is schematically depicted in Figure 1. The objective of this framework is to enable resource-limited devices to solve computationally-expensive problems by redistributing parts of the problem onto other devices. In constructing this grid-based environment, we used the paradigm of a multi-agent system. Each mobile device is viewed as an autonomous intelligent agent [5,9,11], which has a significant degree of autonomy, capable of performing independent tasks, sharing resources with other agents, and communicating with other agents in the grid [8,19]. In this approach, mobile devices work on user-requested tasks in such a way that the user does not know the exact method used to solve the task. A mobile device may be capable of solving the task on its own if it has enough processing power, available memory and battery charge. Otherwise, if the task is too big and/or resource-intensive, the mobile device may delegate portions of the task to other available and willing mobile devices available within the grid-based environment. The environment infrastructure facilitates the distribution of the resource-intensive tasks across the community of other mobile devices connected to the grid

and enables an effective communication and coordination among these mobile devices.

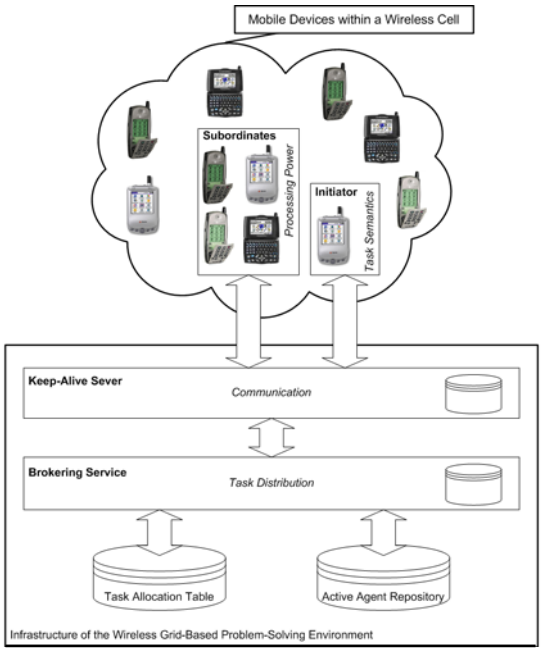

Figure 1. Architecture of the grid-based problem solving devices

Each mobile device that is capable and willing to join the grid-based environment must run a client application implementing a light weight communication protocol for information exchange with the grid's *Brokering Service*. Any mobile device running this client application is called a *Subordinate* upon joining the grid environment. Any Subordinate can be requested to participate in solving a part of a computationally-intensive task for another mobile device. Any Subordinate mobile device may become an *Initiator* of a distributed task if its user starts a large and/or computationally intensive task that cannot be solved by that device itself due to its resource limitations. In this case, it is the Initiator's responsibility to make this task easily distributable across the grid-based environment and to submit this distributed task to the Brokering Service. The Brokering Service and other components of the infrastructure coordinate the distributed execution of the task and facilitate all communication among the devices participating in the grid.

The environment's infrastructure contains two major data stores, the *Active Agent Repository* (AAR) and the *Task Allocation Table* (TAT), which record all information needed by the Brokering Service. AAR contains real-time records about all Subordinates that are currently available to the grid-based environment. This information includes the Subordinate's CPU rating, amount of available physical memory and the current remaining battery charge. TAT contains information describing each distributed task currently running in the environment, including its code and relevant data, how the task is allocated among the Subordinates, and any available partial results. Information stored in TAT dynamically changes as the distributed task gets closer to its completion. When Subordinates complete their respective partial tasks, they submit the results back to the Brokering Service, which stores them in TAT. These results then await collection by the Initiator of the original distributed task.

The *Keep-Alive Server* handles all aspects of communication between the mobile devices and the rest of the grid-based environment infrastructure. It is worth noting that in order to minimize the communication overhead, in our architecture there is no direct communications between individual mobile devices. The Keep-Alive Server sends and receives messages to and from all mobile devices currently participating in problem-solving activities. Mobile devices entering the wireless grid and willing to take advantage of the processing power of other devices and/or to share processing power, must advertise themselves as available to the Keep-Alive Server. A specially-designed mechanism (described in our previous work [11]) is put in place to prevent computing devices from over-using the resources of other mobile devices without making their own resources available to the rest of the grid [10]. As its name suggests, one of essential functions of the Keep-Alive Server is to monitor the availability and current activity of all mobile devices within the grid-based environment, but especially of those Subordinates that are currently employed in solving partial tasks. The Keep-Alive Sever collects real-time information about the progress of each Subordinate towards completing the partial computational task delegated to it by the grid infrastructure. If one of the Subordinates should time-out or abort a partial computational task, the Keep-Alive server will inform the Brokering Service, which will reallocate the corresponding partial task to a different Subordinate.

As illustrated in Figure 1, there is a clear separation between the functionality of the Keep-Alive Server and the Brokering Service based on the nature of their responsibilities. The Keep-Alive Server is only concerned with the communication between mobile devices and environment infrastructure. The Brokering Service has all information about available mobile devices. It knows about the distribution of computational tasks, their current status and the overall progress towards their completion, but not about the task semantics, which are encapsulated within the Initiator. Subordinates are mere workhorses whose processing power is used to solve separate parts of large distributed tasks.

3. Prototype of the grid-based problem solving environment

This section describes our implementation of the grid-based problem solving environment architecture for mobile devices. The main objectives of this prototype are to prove the feasibility of our grid-based architecture, to model all interactions between mobile devices and the environment infrastructure, to prove the concept of task distribution and assembly, and to conduct a number of simulation experiments governed by several control parameters that determine the behavior of the grid.

Grid (or on-demand) computing is a relatively new field of research whose goal is to allow a heterogeneous community of computing devices to share computational resources across the network. A specially designed network infrastructure manages the aggregation of these resources and transparently presents them as a uniform commodity to devices seeking to perform a computationally-intensive task beyond their individual capabilities [3]. A large number of a variety of wired grid architectures developed so far employ SOAP-based XML Web Services to implement grid middleware, which allows future grid implementations to build on the TCP/IP foundation of the global Internet [4].

We developed a prototype for the wireless grid-based environment for mobile devices in order to experimentally determine the effects of device mobility and resource-scarcity on sharing of computational power among wireless mobile devices. The sharing of resources is emulated by the distribution of one or more parallel task, which represent a certain workload that requires more processing power than that available on a single mobile device. In particular, we use the parallel implementation of IDA* algorithm to solve a large constraint satisfaction problem [10]. We chose the task of finding the shortest path on a geographical map because it represents a class of problems that are typical for modern mobile devices. However, given a large and detailed map, this problem may become too complex for some devices, thus necessitating its distribution to other computing devices. The algorithm used to solve such a problem requires neither a large amount of memory for its data (adjacency matrix), nor any intermediate communication or synchronization between the devices used to solve parts of the problem. Thus, partial units of a map routing task are distributed to multiple peer devices for processing. The chosen algorithm guarantees that each of the peer devices will find a feasible solution. Eventually, these results are returned to the initiating device, which may then determine the optimal solution based on partial results representing a set of feasible solutions. Such a problem placed in the domain of handheld computing is also interesting because the user may be satisfied with a sub-optimal result and choose not to wait for all results needed to determine the optimal solution.

The simulation prototype is developed to determine the result of introducing mobility and network instability into the wireless grid-based problem solving environment for mobile devices. A high-level view of the prototype architecture consists of the following three components: 1) client application installed on participating mobile devices, 2) Keep-Alive Server, and 3) Brokering Service. All three components of the prototype are currently implemented using Microsoft .NET Framework. The client application is optimized to run on .NET Compact Framework [21].

3.1. Client application

A light-weight client application is installed on all mobile devices enabling them to access and communicate across the grid based environment. For the purposes of prototyping, the client application is synonymous with the mobile device itself. Mobile devices are characterized by the role they play in the grid, as shown in Figure 1. The role of the mobile device is determined by whether it initiates a task (Initiator) or whether it solves a part of the task initiated by another mobile device (Subordinate). Devices that joined the grid, but not currently serving as initiators or participating in solving a distributed task, are also considered as subordinates. Devices can access different sets of the grid-based environment's functionality based on their current role. Subordinates may request to join or leave the environment, initiate a task (in such a case a Subordinate becomes an Initiator), execute assigned partial tasks or return results of completed partial tasks. Initiators may create a distributed task, abort an initiated task or retrieve results of the completed partial tasks.

3.2. Keep-Alive Server

The Keep-Alive Server is a TCP/IP service that is used for transparent handling of all communication between the grid-based environment (namely, Brokering Service) and the client applications running on mobile devices, as shown in Figure 1. The mobile devices within the grid-based environment communicate with the Keep-Alive Server via the keep-alive protocol at periodic intervals to indicate that they are still active in the grid and are either available to solve distributed partial tasks or to report their progress on solving them, as shown in Figure 2 and Figure 3. Changes to the state of the grid-based environment are reported to the Keep-Alive Server from network devices and from the Brokering Service. The time-sensitive nature of the keep-alive protocol requires that network devices communicate directly with the server to avoid timing-out. However, all time independent changes occur via the Brokering Service, which notifies the Keep-Alive Server of changes made to the state of the grid-based environment by sending custom TCP packets.

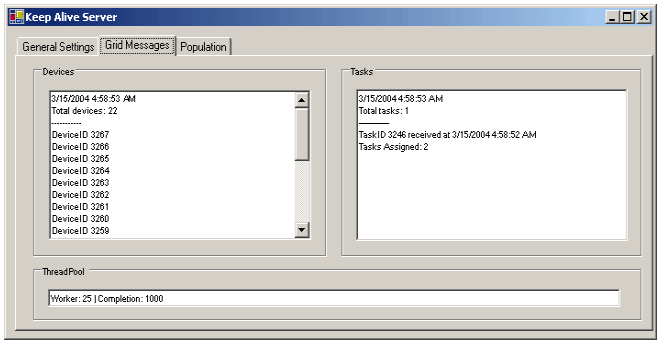

Figure 2. Keep-Alive Server monitors all communication aspects of the grid

3.3. Brokering Service

The Brokering Service is the central part of the grid-based environment. It is implemented as a suite of XML Web Services that exposes methods used to collect and store detailed data, which describe the characteristics of the mobile devices comprising the environment population. It also contains information about the current state of the tasks initiated and distributed across the grid-based environment. This information is maintained in the AAR and the TAT, respectively. In particular, the AAR stores information describing various characteristics of each mobile device currently registered with the environment, which includes the device type, role (Subordinate/ Initiator), processor speed rating and the amount of physical memory. The TAT contains information describing distributed tasks, which includes a list of arguments required for task execution, the identity of the Initiator and the executable byte-code of the task. Updates made to either the AAR or TAT are propagated via TCP packets sent by the Brokering Service to the Keep-Alive Server, which is kept current of all changes to the state of the grid-based environment.

When a mobile device requests to join the environment, the characteristics of this device are entered into the AAR. Changes in device's characteristics are reported to the Brokering Service via the Keep-Alive protocol, in which case the AAR is updated with the new information. This ensures that the Brokering Service is capable of efficiently determining an efficient distribution for any initiated tasks.

If a mobile device registered with the environment initiates a new distributed task, the source code needed to run the corresponding partial tasks and all relevant parameters are submitted to the Brokering Service, which dynamically compiles the code into an executable segment. This data, as well as the identity of the Initiator, is recorded into the TAT. Following that, the Brokering Service assigns the newly submitted task to available Subordinates. The Keep-Alive Server is then notified that a task is ready to be distributed and transmits the task to the appropriate Subordinates specified by the Brokering Service. Since the work described here presents prototyping and simulation of the grid-based architecture, we deliberately made no attempt of certifying the source code deployed from the Initiator. However, this can be achieved by using digital certificates issued by trusted certifying agencies. Figure 4 illustrates the algorithm used to

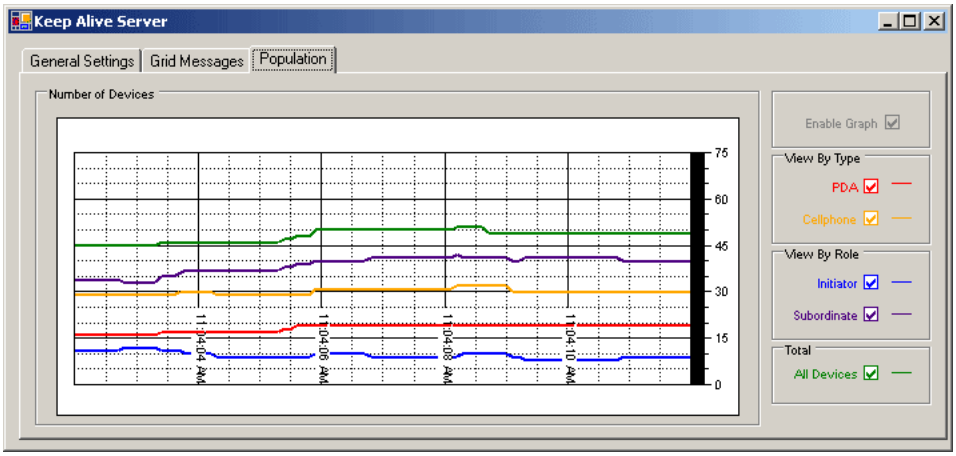

Figure 3. Keep-Alive Server monitors all mobile devices comprising the population of the grid

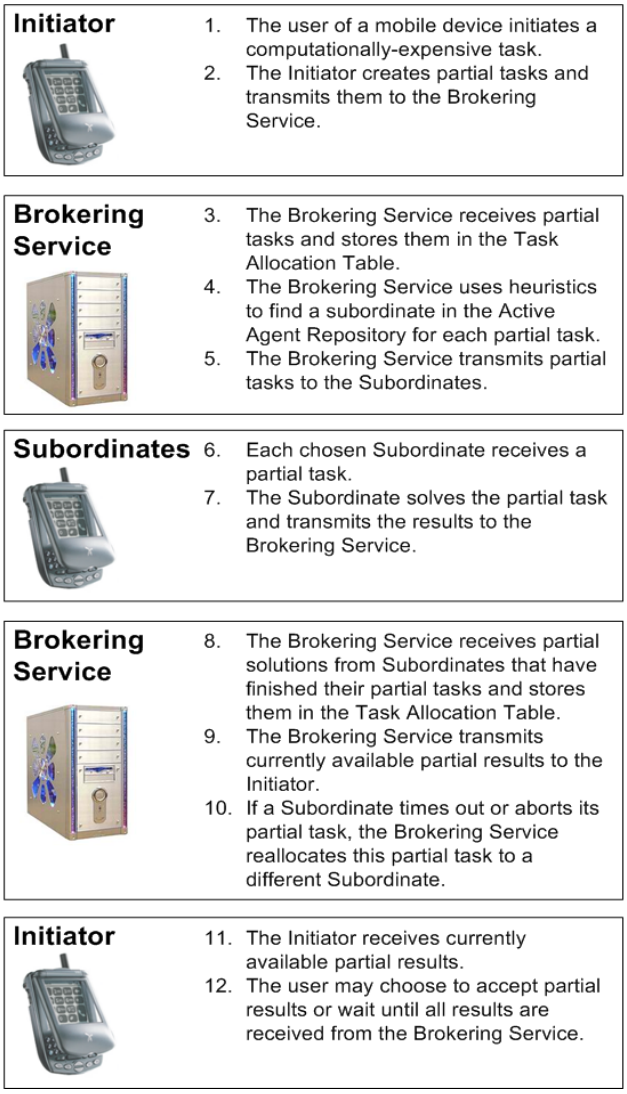

Figure 4. Task distribution algorithm used in the grid-based problem solving environment

a dependable system in itself or part of the underlying dependable network. For applications such as financial transactions and mobile inventory management, network dependability is critical. Unexpected

distribute and solve a computationally-expensive task in the grid-based problem solving.

4. Issues of transparency and dependability

Transparency in a computer system implies hiding irrelevant details of implementation from the users of the system. Transparent access to grid-based aggregation of computing power grants mobile devices with limited resources the means to obtain results of computationally-intensive tasks by harnessing the collective resources of the grid. Because it shields the user of the mobile device from the system's complexity, the proposed grid-based model provides transparent access to shared resources. A popular method to ensure transparency of access is to support the intermittent connectivity suffered intrinsically by wireless environments. If intermittent connections are not supported, the user becomes aware of the underlying network and connectivity issues [20]. Planning for and recovering from sporadic connectivity provides a seamless and transparent experience to the users. The grid-based problem solving model attempts to encapsulate the complexity of the underlying architecture from clients of the resource-sharing services.

Closely linked to the issue of transparent access is the issue of seamless handoff [7]. For example, in the proposed model, if the Subordinates or the Initiator moves out of range of the wireless cell containing the original Brokering Service, then handoff mechanisms in place ensure a smooth, transparent transition to the next cell. The user is unaware of the change of control. However, if soft and seamless handoffs are not employed, the user unnecessarily becomes aware of the underlying complexity of the wireless grid. In a heterogeneous environment of different types of devices running different software, participation in the grid necessitates effective handling of the issue of transparent access.

Latency in handoffs can lead to data losses in the wireless grid. In a resource-sharing environment such as the grid-based problem solving model proposed here, loss of data can result in potentially erroneous outcomes. When the Initiator requests thatpartial tasks be distributed by the Brokering Service, the Subordinates need to obtain correct data in a timely manner to complete the sub-tasks and return partial results to the Initiator so that they are useful and arrive in a well-timed fashion. Otherwise, the usefulness of the grid will be limited as users will be hesitant to share tasks in a latent and error-prone environment. Users also need reliable indicators of availability of the grid services.

According to [14], dependability can be defined as the trustworthiness of a computing system. If a system is dependable, we can rely on its service. Reliability of the system, availability of services, safety and security of the network are, therefore, attributes of dependable systems. While the wireless grid can be used for computationally-intensive tasks, it must also be idisconnections common in the wireless environment of the grid adversely impact the dependability of the system and limit its usefulness.

How frequently the components in the grid fail will undermine the availability of services. Since the users desire constant uptime, the grid-based environment has to be designed with a goal toward maximizing fault tolerance. The security of mobile transactions across the grid can also be impacted by frequent component failure [20]. The mobility of the devices is a critical factor affecting grid performance. In other words, if the devices in the grid are highly mobile, this adversely impacts the time taken to perform a computationally-intensive task. A related issue is that of Subordinates and/or the Initiator leaving the cell while partial tasks are still in progress. If the Subordinates involved in working on partial tasks leave the grid, the Brokering service re-allocates these partial tasks to other Subordinates present in the grid. When the Initiator of a task moves to a new area, the Brokering Service is currently programmed to abort all partial tasks associated with the Initiator.

Grid dependability is directly correlated with Quality of Service (QoS) issues. The long-term goal of the wireless grid is to provide the same QoS to mobile users as experienced by wired users. High reliability and fault-tolerant design in the grid-based model facilitate in improving wireless QoS. Due to the inherent characteristics of wireless media, frequent disconnections occur, which may result in poor QoS. Treating frequent disconnection as an anticipated fault and recovering from the adverse effects enables the resource-sharing grid to be dependable in providing services to mobile devices. One efficient method to increase dependability is to have redundant components for critical infrastructure in the grid. If users are switched to a standby component in case of a fault, then dependability levels are increased and availability to grid-based services or network uptime is maintained at pre-fault levels.

The two major factors that can cause network dependability to suffer are improper flow control and congestion control [7]. If steps are not taken to include redundant components in case of congestion, then availability and reliability are adversely impacted. If the flow of data to the Subordinates and the flow of partial results from Subordinates to the Brokering Service are not managed, then security and fault-tolerance are affected. With the assistance of the Keep-Alive Server and synchronous management of the Brokering Service with the mobile devices, the proposed model manages the flow of data in both directions. Also, alternate routes should be designed in case of congestion.

According to the classification of grid applications mentioned in [17], pipelined and synchronized applications are computationally-intensive and are distributed across devices participating in the grid. An example application of this class can be considered to be the parallel IDA* search algorithm that we used to prove that a wireless grid-based environment for can be used as a problem solving collaborative framework among resource-scarce mobile devices. This algorithm can be solved in parallel, it lends itself to a sub-optimal solution scenario, and it is computationally-intensive, so that an individual mobile device cannot solve it on its own. Experiments of the grid-based environment functionality using this algorithm are ongoing and results are expected shortly.

5. Preliminary results and future work

The prototype of the grid-based problem solving environment is designed to allow for the experimentation with the values of three control variables: initial grid population size, device mobility and frequency of task initiation (Figure 5). The initial grid population size specifies the number of mobile devices in the environment at the beginning of a simulation run. Device mobility indicates the probability per unit of time of the event that a new mobile device joins or an existing device leaves the environment. The frequency of task initiation indicates the probability per unit of time of the event that a device initiates a new distributed task (that is, a Subordinate becomes an Initiator). The variable used to measure the efficiency of the grid implementation is the average time taken for a task to be initiated, distributed, solved by the Subordinates and returned back to the Initiator.

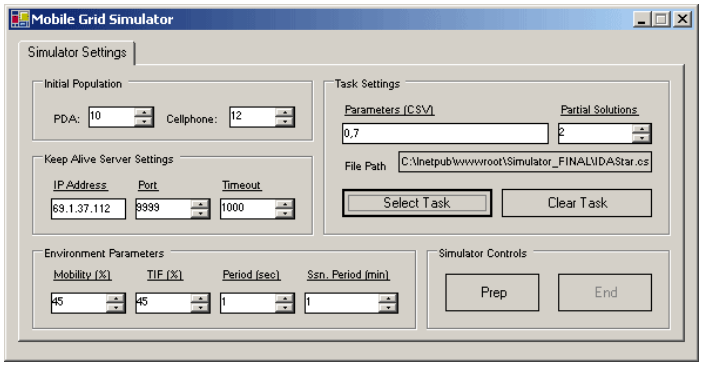

Figure 5. Grid-based architecture prototype allows changing many simulation parameters

Our plans for future work include conducting a wide variety of simulation experiments using various scenarios determined by different combinations of the control variables. Using different combinations of the three control variables will result in at least twelve distinct simulation scenarios. Each scenario is determined by fixed values of two out of the three control variables, while the third variable is subject to changes in its value. We are particularly interested in the effects of network instability and high device mobility on the overall performance of the grid measured by the task completion time. Specifically, we plan to focus our experiments to determine the varying impact of network instability on the grid performance in situations with different device populations.

6. Summary

In this paper we presented a framework modeling a collaborative problem solving environment for mobile devices. It can be used for sharing of processing resources across several mobile devices to redistribute the computational load of a large and/or complex computing problem that a standalone mobile device would not be able to complete by itself. The currently implemented simulation of this framework is aimed at studying the effects of different network configurations taking into account the device mobility, task initiation frequency and the number of available mobile devices. We expect the simulation results to demonstrate the advantages of the proposed framework for collaboratively solving computationally-intensive tasks.

7. References

1. Allen, G. et al. (2003). Enabling Applications on the Grid: A GridLab Overview. *International Journal of High Performance Computing Applications: Special issue on Grid Computing: Infrastructure and Applications*, 17(04) 449- 459.

2. Bhagyavati, Kurkovsky, S (2003). Wireless Grid Enables Ubiquitous Computing. *Proceedings of The 16th International Conference on Parallel and Distributed Computing Systems (PDCS-2003)*, Reno, NV.

3. Foster, I. et al. The Anatomy of The Grid: Enabling Scalable Virtual Organizations. 171-197. *In:* Berman, F., Fox, G.C., Hey, A.J.G. (Eds.), *Grid Computing*, Wiley.

4. Foster, I. et al. (2003). The Physiology of the Grid. 217-249. In Berman, F., Fox, G.C., Hey, A.J.G. (Eds.), Grid Computing, Wiley.

5. Fukuda, M. et al. (2003). A Mobile-Agent-Based PC Grid. *Proceedings of The 5th Annual International Workshop on Active Middleware Services (AMS2003)*, Seattle, WA.

6. Ynor, M., McKnight, L.W., Hwang, J. and Freedman, J. (2003).Wireless Grid Networks and Virtual Markets.*Proceedings of International Conference on Computer, Communication and Control Technologies (CCCT '03)*, Orlando, FL.

7. Hsieh, H., Kim, K., Zhu, Y. Sivakumar, R. (2003). A Receiver-Centric Transport Protocol for Mobile Hosts with Heterogeneous Wireless Interfaces. *Proceedings of The 9th International Conference on Mobile Computing and Networking (MobiCom'03)*, San Diego, CA.

8. Jennings, N.R., Sycara, K., and Wooldridge, M. (1998). A Roadmap of Agent Research and Development. 7-38. *In*: *Autonomous Agents and Multi-Agent Systems Journal*, Jennings, N.R., Sycara, K. and Georgeff, M. (Eds.), Kluwer Academic Publishers, Boston, MA, 1(1).

9. Kuang, H., Bic, L. Dillencourt, M. (2002). Iterative Grid-Based Computing Using Mobile Agents. *Proceedings of the 2002 International Conference on Parallel Processing*, Vancouver, B.C., Canada.

10. Kurkovsky, S. Bhagyavati (2003). Agent-Based Distributed IDA* Search Algorithm for a Grid of Mobile Devices. *Proceedings of The 7th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI-2003)*, Orlando, FL.

11. Kurkovsky, S. and Bhagyavati (2003). Modeling a Computational Grid of Mobile Devices as a Multi-Agent System.*Proceedings of The 2003 International Conference on Artificial Intelligence (IC-AI'03)*, Las Vegas, NV.

12. Li, Z., Wang, C. and Xu, R. (2001). Computation Offloading to Save Energy on Handheld Devices: A Partition Scheme. *Proceedings of International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES'01),* Atlanta, GA.

13. McKnight, L.W, Howison, J. (2003). Towards a Sharing Protocol for Wireless Grids. *Proceedings of International Conference on Computer, Communication and Control Technologies (CCCT '03)*, Orlando, FL.

14. Parhami, B. (1998). From Defects to Failures: a View of Dependable Computing. *ACM SIGARCH Computer Architecture News – Special Issue on Architectural Support for Operating Systems*, 16(4), ACM Press, New York, NY.

15. Phan, T., Huang, L. and Dulan, C. (2002). Challenge: Integrating Mobile Wireless Devices into the Computational Grid. *Proceedings of The 8th International Conference on Mobile Computing and Networking (MOBICOM'02)*, Atlanta, GA.

16. Ray, A. and Bhagyavati. (2003). Mobile Devices in a Computational Grid. *Proceedings of The 41st ACM Southeast Regional Conference (ACMSE-03)*, Savannah, GA.

17. Snavely, A. et al. (2003) Benchmarks for Grid Computing: A Review of Ongoing Efforts and Future Directions. *ACM SIGMETRICS Performance Evaluation Review,* 30(4), ACM Press, New York, NY.

18. Sterbenz, J.P.G. et al. (2002). Survivable Mobile Wireless Networks: Issues, Challenges, and Research Directions. *Proceedings of the ACM Workshop on Wireless Security*, Atlanta, GA.

19. Tomarchio, O. and Vita, L. (2001). On the Use of Mobile Code Technology for Monitoring Grid System. *Proceedings of First IEEE/ ACM International Symposium on Cluster Computing and the Grid (CCGrid 2001)*, Brisbane, Australia.

20. Varshney, U. and Vetter, E. (2002). Mobile Commerce Framework, Applications and Networking Support. *Mobile Networks and Applications*, 7(3)

21. Wigley, A, Wheelwright, S. (2002). *Microsoft .NET Compact Framework*. Microsoft Press.