ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2012, Vol. 41, No. 4

An Adaptive and Distance-based Resource Allocation Scheme for Interdependent Tasks in Mobile Ad Hoc Computational Grids

Sayed Chhattan Shah[1], Myong-Soon Park[2], Wan Sik Choi[1], Zeeshan Hameed Mir[1], Sajjad Hussain Chauhdary[3], Ali Kashif Bashir[4], Fida Hussain Chandio[5]

[1] *Electronics and Telecommunications Research Institute, Daejeon, Korea*
e-mail: shah@etri.re.kr

[2] *Department of Computer and Radio Communications Engineering, Korea University, Seoul, Korea*
e-mail: myongp@korea.ac.kr

[3] *LS Industrial Systems, Korea*
e-mail: sajjad@lsis.biz

[4] *National Fusion Research Institute, Daejeon, Korea*
e-mail: ali12@nfri.re.kr

[5] *Institute of Mathematics and Computer Science, University of Sindh, Pakistan*
e-mail: fida.chandio@usindh.edu.pk

crossref http://dx.doi.org/10.5755/j01.itc.41.4.877

Abstract. Two key components contribute to task completion time: execution cost and communication cost. The communication cost is induced by data transfers between tasks residing on separate nodes. The communication is always expensive and unreliable in mobile ad hoc Grids and therefore plays a critical role in application performance. To reduce communication cost, interdependent tasks are allocated to nodes located close to one another. However, once the tasks have been allocated, nodes can move within a Grid. The movement of nodes within a Grid may result in multi-hop communication between nodes executing dependent tasks. In order to deal with node mobility within a Grid, an effective resource allocation scheme is required, but the design of such a scheme for mobile ad hoc computational Grids is challenging due to the constrained communication environment, node mobility, and infrastructure-less network environment. In this paper, we have developed an adaptive and distance-based resource allocation scheme which takes into account the characteristics of an application and nodes and applies migration heuristics to address the local node mobility problem. The scheme is validated in a simulated environment using various workloads and parameters.

Keywords: Computational Grid; Mobile Grid; Ad Hoc Networks; Resource Allocation; Interdependent Tasks.

1. Introduction

Due to recent advances in mobile computing and communication technologies, mobile ad hoc computational Grids are emerging as a new computing paradigm, enabling innovative applications through the sharing of computing resources among mobile devices without any pre-existing network infrastructure. Mobile ad hoc computational Grids are integration of computational Grids and mobile ad hoc networks. A computational Grid is a software infrastructure that allows distributed computing devices to share computing resources to solve computationally intensive problems [3], while a mobile ad hoc network is a wireless network of mobile devices that communicate with one another without any pre-existing network infrastructure [1].

The idea of the mobile ad hoc computational Grid is motivated by recent advances in mobile computing and communication technologies which now make it feasible to design and develop the next generation of applications through the sharing of computing resources in mobile ad hoc environments. For example, members of a group of miniature autonomous mobile robots deployed in an urban environment can collaborate with each other to perform an automated video surveillance task, or soldiers in a group can use their wearable computing devices and range of sensor nodes to form a Grid in order to construct a 3D map

and to identify and monitor stationary and moving objects within a map.

This paper addresses the problem of resource allocation to interdependent tasks in mobile ad hoc computational Grids. The task dependencies can be classified into two categories: control dependencies and data dependencies. With data dependencies, tasks exchange data with one another in order to achieve desired results. The data dependencies between tasks imply that heavy communication can be induced by data transfers between tasks residing on separate nodes. Communication is always expensive and unreliable in mobile ad hoc Grids and therefore plays a critical role in application performance. To reduce the communication cost, interdependent tasks are allocated to nodes that are closer to one another. However, once the tasks have been allocated, the nodes can move within the Grid, and this movement of nodes within the Grid may result in multi-hop communication between nodes executing dependent tasks. In order to deal with node mobility within a Grid, an effective resource allocation scheme is required, but the design of such a scheme for mobile ad hoc computational Grids is challenging due to node mobility, the constrained communication environment, limited battery power, and the infrastructure-less network environment.

Problems due to node mobility: Unpredictable node mobility across the coverage area may result in task failure, and within the coverage area it may increase communication cost. Node mobility across the coverage area affects not only tasks executed on nodes but also dependent tasks executed on other nodes. Moreover, in a multi-hop mobile ad hoc Grid, a node can also be used as an intermediate node to forward data on behalf of tasks executed on neighboring nodes. Therefore, the mobility of a single node can have an enormous effect on application performance.

Problems due to the constrained communication environment: A mobile ad hoc network provides networking and communication services to nodes within a Grid. It presents a very constrained communication environment due to the limited power, shared medium, available spectrum, and node mobility, and hence suffers from limited bandwidth, high latency, and unstable connectivity problems, which may result in severe network congestion due to frequent failure and activation of links [1]. In such an environment, data transfer between dependent tasks is very critical for task completion time.

Problems due to battery power: Nodes within a Grid are battery-driven and their power is limited and should be utilized effectively to prolong the lifetime of the nodes and thus of the application. Ineffective allocation of tasks to mobile nodes can significantly increase the communication and energy consumption cost, which limits the lifetime of nodes and may result in power failure. The power failure will affect not only the task executed on a node but also dependent tasks executed on neighboring nodes.

In the literature, various schemes have been proposed to address the resource allocation problem, but most of them are either targeted towards pre-existing network infrastructure-based systems [11], [27] or they do not consider dependencies between tasks [4], [8], [36], [26].

In our previous studies, we have proposed two resource allocation schemes: (1) a centralized and distance-based resource allocation scheme [29] which exploits the characteristics of an application and allocates interdependent tasks to nodes located close to one another to reduce the cost of communication between interdependent tasks; and (2) an energy efficient resource allocation scheme [30] which exploits a transmission power control mechanism to reduce energy consumption in the transmission of data. However, neither of these schemes are adaptive to node mobility within a Grid coverage area.

In this study, we have developed an adaptive and distance-based resource allocation scheme which takes into account the characteristics of an application and nodes and applies migration heuristics to reduce the communication cost and energy consumption in the transmission of data. The scheme is based on a group mobility model in which all nodes work as a group and is adaptive to node mobility within a Grid. The scheme is validated in a simulated environment using various workloads and parameters.

2. Related work

Most of the work on resource allocation in computational Grids is focused towards infrastructure-based powerful computing systems connected through high performance communication networks [2], [5], [6], [7], [10], [19], [22], [28], [35], [32]. However, due to recent advances in mobile computing and communication technologies, there is a significant shift towards mobile Grids research. The research on mobile Grids is divided into two categories. In the first category, mobile devices are allowed to access Grid resources [12], [15], [17], [23], [25], while in the second category they can be used as a computing resource within a Grid [13], [14], [31]. The second category is further divided into two subcategories: in the first, mobile devices are integrated with infrastructure-based computing systems [9], [16], and in the second they can collaborate with each other without any pre-existing network infrastructure [18], [32], [33]. The latter is referred to in this paper as a mobile ad hoc computational Grid.

The research on resource allocation in mobile ad hoc computational Grids is still in a preliminary phase and a very few schemes based on a decentralized architecture have been proposed to address issues such as node mobility, energy management, and task failure. For example, Hummel and Jelleschitz [18] proposed a distributed resource allocation scheme

based on a first-come-first-served strategy that allows each mobile node to perform mapping based on the job's requirements. It employs a proactive and reactive fault tolerance mechanism and supports redundant execution of tasks to deal with task failure. The scheme proposed by Chu and Humphrey [9] utilizes a manager–worker model to distribute tasks and supports application-controlled migration to deal with failure due to low battery power. The problem of energy-constrained scheduling for Grid environments has been addressed by Li and Li [20], who have investigated energy minimization and Grid utility optimization problems. To maximize the Grid system utility without exceeding the deadline and total energy budget, they employ a pricing-based decomposition method. The scheme proposed by Liu et al. [21] also focuses on a power-aware allocation to support the adaptation needs of ad hoc applications such as changes in network topology and application behavior. To reduce the mean path length of data packets, tasks are migrated to topologically closer nodes; however, the migration occurs after analyzing data communication patterns during the execution of tasks. To select the most suitable node for task execution, Gomes et al. [14] proposed a scheme which utilizes a delayed reply mechanism in which a more resourceful node replies earlier than less resourceful nodes. It also provides load balancing and scalability. The scheme proposed by Selvi et al. [33] addresses node mobility by profiling the regular movements of a user over the time. The profiling consists of the user's visited locations and associated time spent at those locations. A node which previously stayed longer at a location is selected for task execution.

The schemes mentioned above are based on a decentralized architecture that results in poor allocation decisions due to the lack of a network-wide view. They also do not consider the dependencies between tasks and are targeted towards load balancing, scalability, and fault tolerance rather than application performance. Moreover, these schemes also do not address the node mobility problem within a Grid coverage area. To deal with precedence dependencies, Shilve at al. [32] have proposed a scheme based on static allocation of resources in ad hoc computational Grids; however, due to static allocation, this scheme is not adaptive to network changes and application behavior.

3. System models

This section is further divided into two subsections: the first subsection describes the network model and the second the application model.

3.1. Network Model

A mobile ad hoc network is modeled as an undirected graph $G_N = (N, L, p, m, b)$, where N is the set of vertices representing mobile nodes and L is the set of edges representing communication links among them. The parameters p_i, m_i, and b_i represent the processing power, memory, and remaining battery power of node n_i, respectively, while D_{ij} represents the communication distance between nodes n_i and n_j connected by a link l or set of links, where $l \in L$. The nodes can move within and across the network coverage area and are heterogeneous in terms of processing power, memory, and battery power.

3.2. Application model

A parallel application is modeled as a graph $G_A=(T, C)$, where T is the set of vertices representing tasks and C is the set of edges representing dependencies between tasks. The tasks within an application are preemptive and indivisible work units. The dependencies are divided into two categories: precedence dependencies and parallel execution dependencies. The tasks with precedence dependencies are executed independently but require inputs generated by predecessor tasks, while tasks with parallel execution dependencies periodically exchange data with one another and communication between tasks may take place at any time during execution. The precedence dependency of task t_j on task t_i implies that task t_i must be completed before task t_j, while the parallel execution dependency of task t_i on task t_j implies that the execution of both tasks should start at the same time.

In addition to dependencies, tasks are also divided into three categories: computation-bound tasks, local communication-bound tasks, and remote communication-bound tasks, represented by $t_i^{cpu-bound}$, $t_i^{lc-bound}$ and $t_i^{rc-bound}$, respectively. The computation-bound tasks exchange a small quantity of data and have high processor utilization, while communication-bound tasks exchange a large quantity of data and have low processor utilization. Among communication-bound tasks, local communication-bound tasks spend most of the time performing local I/O operations while remote communication-bound tasks spend most of the time performing remote I/O operations.

The purpose of classifying dependencies and tasks is to exploit them in order to improve the utilization of computing resources and application performance. For example, in the case of computation-bound tasks, high processing nodes are critical for their performance, while in the case of communication-bound tasks, communication performance is more critical than processor performance. Moreover, among communication-bound tasks, remote communication-bound tasks are more critical to performance than local communication-bound tasks. Like tasks, knowledge of dependencies also plays a key role in improving application performance and resource utilization. For example, communication between tasks with parallel execution dependencies may take place anytime during execution; therefore, such tasks

should be allocated simultaneously or with the minimum possible delay. Otherwise, it is possible for one task to be allocated and waiting for data from another task which is still awaiting allocation. In this situation, the allocated task would not be able to proceed and would be wasting valuable resources.

4. An adaptive and distance-based resource allocation (ADRA) scheme

ADRA aims to reduce the communication cost between interdependent tasks. It takes into account the task and dependency types and allocates interdependent tasks to nodes which are close to one another with respect to physical distance. The use of physical distance as a metric can result in a better performance when nodes use multiple transmission power levels to communicate with each other [30]. This is because tasks executed on two nearby nodes accessible at minimum transmission power do not require maximum transmission power to communicate, which can significantly reduce energy consumption and communication cost. However, for nodes with fixed transmission power, the distance is measured in numbers of hops.

4.1. Resource Allocation

This section is divided into two parts: The first part focuses on the node selection mechanism and the second part describes the resource allocation method.

Node selection mechanism

In order to select nodes for allocation of tasks we adopt an approach proposed in [34] which is used to predict the amount of time during which two nodes will remain connected to each other. It is assumed that each mobile node is equipped with WA-DGPS, which provides position, speed, and direction information. Nodes share this information with each other in order to predict the future connectivity. For details, readers are referred to [34]. The nodes that will remain connected for a longer period of time are selected for allocation of tasks. The node selection mechanism is used in scenarios where the movement of one node is independent from that of the others.

Task allocation

Before allocation, all tasks are sorted and are assigned to different levels depending on precedence and parallel execution dependencies. The lowest level consists of tasks with no predecessors and the highest level consists of tasks with no successors. Tasks with parallel execution dependencies are assigned to the same level. At each level, tasks are assigned a priority according to task type. The remote communication-bound tasks have the highest priority, followed by local communication-bound and computation-bound tasks. Allocation starts from the lowest level, and at each level, tasks are considered by priority. Only tasks with no predecessors or whose predecessors have completed their execution are considered for allocation. In order to make allocation decisions, there are three possible cases:

Allocation of an independent task: Since an independent task does not have dependency, the resource allocation service makes an allocation decision according to the task type. For local and remote communication-bound tasks, a low processing node is selected, while for computation-bound tasks, a high processing node is selected for task execution.

Allocation of interdependent tasks set T^*: A set of interdependent tasks consists of multiple tasks with parallel execution dependencies. For allocation, one remote communication-bound task $t_i^{rc-bound}$ is selected from the interdependent tasks set T^* and is allocated to a low processing node. The remaining tasks are allocated close to this task.

Allocation of dependent tasks T^* that have dependency on an already allocated task t_r: One task is already allocated while other dependent tasks are waiting for allocation. The allocated task is called a reference task t_r while the node that is executing a reference task is called a reference node n_r. Dependent tasks T^* are allocated close to task t_r.

4.2. Adaptation Mechanism

Once the tasks have been allocated, the nodes can move within a Grid. Node mobility within a Grid can increase the communication distance and may result in multi-hop communication between nodes executing dependent tasks. Multi-hop communication increases queuing and packet processing delays, the number of forwarded, dropped, and lost packets, and the amount of control traffic. It may also generate a new set of control packets due to route rediscovery and medium access control. In order to avoid multi-hop communication between dependent tasks due to local node mobility, we have developed an adaptation algorithm which migrates dependent tasks to nearby nodes.

In this study we have assumed that the amount of data transmitted or processed by tasks is unknown. Without this assumption, it would be easy to make an effective migration decision by estimating the task completion time before and after migration of the task. Since the amount of data transmitted or processed by tasks is not known in advance, we have to exploit the application's characteristics, such as task and dependency type.

This section first describes key factors that should be considered when making a migration decision and then lists the migration heuristics.

Factors critical to the migration decision

Type of task: It is very important to consider the type of task executed on a mobile node. As defined in the application model, remote communication-bound tasks exchange a large quantity of data and therefore are more critical to task performance than local communication-bound or computation-bound tasks.

Computation-bound tasks, however, exchange small amounts of data, so there is a high probability that migration of computation-bound tasks would not result in better performance.

In addition, a task can process data stored on a node where the task is executed or collected from an external environment. In the former case, it is important to consider the amount of data that need to be migrated with the task. In the latter case, data are collected from the environment, so migration of the task close to its dependent tasks would result in better performance.

Type of dependency: In the case of precedence dependency, a successor task communicates only once to collect results, while in the case of parallel execution dependency it is likely that dependent tasks will continue to exchange data throughout their execution. The quality of the connection between nodes may vary significantly with respect to time. For instance, the quality of the connection between nodes executing dependent tasks may be the best at the start, while later it may deteriorate. In the case of precedence dependency this issue is not serious because the predecessor task communicates only once to collect results. In the case of parallel execution dependency, the connection quality between nodes should be monitored carefully so that migration decisions can be made.

Number of mobile nodes: All the nodes executing the interdependent task set may move together or some of them may move while others may remain stationary. In such a situation the task type and numbers of mobile and stationary nodes are critical for an effective decision. When the majority of nodes executing interdependent tasks are moving while a few of them are stationary, it is better to migrate tasks executed on stationary nodes near to mobile nodes. Otherwise, migration of tasks from mobile nodes close to stationary nodes may result in a better performance.

Node direction: Nodes executing dependent tasks can move in the same direction or opposite directions. The former is not a problem, but in the case of the latter, the distance between nodes executing dependent tasks would increase and eventually result in multi-hop communication.

Communication distance: The movement of nodes may increase or decrease the communication distance between nodes executing dependent tasks. A decreased communication distance is better for communication performance. However in the case of increased communication distance, the difference in distances, that is, the new distance minus the old distance, could be small, with a minor effect on communication cost, or large, with a significant effect.

Migration heuristics

- Tasks are migrated when the increased communication distance is greater than a threshold value. A small increase in communication distance usually does not increase the communication cost. In addition, the distance threshold would avoid unnecessary migration.
- We only consider migration of remote communication-bound tasks. As mentioned earlier, remote communication-bound tasks exchange large quantities of data and are therefore more critical to communication performance.
- Computation-bound and local communication-bound tasks are not considered for migration. These tasks exchange small amounts of data and thus do not have a significant effect on communication performance [27]. We assume that the time required to move a local communication-bound or computation-bound task from a distant node to a node located nearby is greater than the time required for data transmission from a distant node.
- We also take into account the number of mobile nodes executing remote communication-bound tasks during a pre-defined time interval. This ensures that if more than one node executing dependent tasks moves within a short interval then all nodes will be treated at the same time.
- Tasks which collect data from the environment are always migrated.

For details refer to the adaptation algorithm.

5. Simulations and analysis

The performance of the proposed scheme (ADRA) is compared with a distance-based resource allocation (DRA) scheme [31] and DICHOTOMY [14]. DRA allocates interdependent tasks to nodes located nearby, while DICHOTOMY utilizes a delayed reply mechanism to select nodes in which more resourceful nodes reply earlier than less resourceful nodes. Both schemes are not adaptive to node mobility within a Grid.

5.1. Performance Metrics

Since an average end-to-end communication delay is a key component that determines communication performance, it is used as a basic performance metric for evaluation of the proposed scheme. In addition, the accumulative application completion time and energy consumption are also used as performance metrics.

Average end-to-end communication delay: This refers to the time taken for a packet to be transmitted across the network from source to destination. This includes all possible delays such as transmission delay, propagation delay, packet processing delay, queuing delays, and so on.

Accumulative application completion time:

$$A_{ACompTime} = \sum_{i}^{n} T_{CompTime}^{i} \quad (3)$$

/* $M^* \subseteq T$, which maintains a list of tasks with updated positions during time interval I , $Q_i^* \subseteq M^*$ represents tasks with parallel execution dependency on t_i , $R^* \subseteq T^*$ is a subset of remote communication-bound tasks , and $S^* \subseteq R^*$ remote communication-bound tasks on a stationary node */

If $t_i^{rc-bound}$ locationUpdated and T_i^* != null **then**

Wait for time interval I

$t_i^{rc-bound} \cup M^*$

Check updated position of tasks having parallel execution dependency on task $t_i^{rc-bound}$

For each task $t_i^{rc-bound} \in M^*$

Find Q_i^*

If Q_i^* = null and updatedDistance of $t_i > d_d$ **then**

Select $t_j^{rc-bound} \in T_i^*$

allocateTasks ($t_j^{rc-bound}$, $t_i^{rc-bound}$)

Else Q_i^* != *null* & updatedDistance of $t_i^{rc-bound} > d_d$

$Q^* \leftarrow Q_i^* \cup t_i^{rc-bound}$

If $Q^* = R^*$ **then**

Get $t_j^{rc-bound} \in Q_i^*$

allocateTasks ($t_j^{rc-bound}$, Q^*)

Else If $Q^* <= S^*$ **then**

Find $t_s^{rc-bound} \in S^*$

allocateTasks ($t_s^{rc-bound}$, Q^*)

Else

Find $t_q^{rc-bound} \in Q^*$

allocateTasks ($t_q^{rc-bound}$, S^*)

allocateTasks(t_r , T^* **)** {

Get a reference node n_r from the task allocation table

Find the closest node n_n from a reference node n_r

Do {

Find candidate node n_c /* Distance between n_c and n_r <= Distance between n_n and n_r */

Assign weight and add n_c to candidate node list N^*

} **Repeat** (until all candidate nodes are found)

Sort N^* based on W_i in descending order

Sort T^* based on task type

For each task t_i within task set T^*

Allocate task t_i to node $n_j \in M^*$

}

Pseudo code of adaptation algorithm

Energy consumption: This is the amount of energy consumed in the transmission of data.

5.2. Simulation Setup

The network simulator NS2 was used for performance evaluation with a wide range of tasks and mobility scenarios. The environment was designed in compliance with a battlefield scenario. A group mobility model [24] was used in which all nodes work as a group. Twenty-four nodes were deployed randomly in subgroups of varying size. The nodes within a group moved to a random destination at a speed distributed between two and four meters per second, and when they reached the boundary of the group, they bounced back and continued to move to another randomly selected destination.

To emulate the varying number of tasks and communication between them, constant bit rate applications were deployed. The amount of data transferred between tasks, which also reflects the task size, was distributed according to the type of task. For example, remote communication-bound tasks exchanged more data than local communication-bound and computation-bound tasks. In order to make allocation decisions, three key services were implemented: a monitoring service, a discovery service, and a resource allocation service. The monitoring service runs on nodes willing to share computing resources while the resource-allocation and -discovery services are executed on a node that requires additional computing resources. The node-monitoring service listens for requests to share computing resources and keeps track of task execution. The resource-discovery service broadcasts request messages and collects replies while the resource-allocation service makes allocation decisions. For a detailed description of these services, the reader is referred to [10]. In order to achieve confidence in the results, each experiment was repeated four times. The parameters specific to scenarios are described in the respective sections, while simulation parameters are given in Table 1.

Table 1. Simulation parameters

Simulation time	1000 seconds
Number of nodes	24
Transmission range	250 m
Simulation area	1500 m × 1000 m
Number of tasks	12, 24, 36, or 48
Transport protocol	TCP
Ad hoc routing protocol	AODV and DSR
Mac protocol	IEEE 802.11
Traffic type	Constant bit rate
Packet rate	2–4 packets per second
Packet size	512 bytes

5.3. Simulation Results

This section, which is divided into five subsections, presents the simulation results.

5.3.1. Effect of node mobility on average end-to-end communication delay

Scenario 1: The ratios of remote communication-bound tasks to local communication-bound and computation-bound tasks are 2:1 and 2:1, respectively. The ratio of tasks with parallel execution dependencies to tasks with precedence dependencies is 2:1. Nodes were moving at a speed of two meters per second. A total of 24 tasks were deployed on 24 nodes and the distance was measured in number of hops. For simplicity, we assumed the same execution cost for all tasks because we are only concerned with the communication performance of an application.

Figure 1. demonstrates an average end-to-end communication delay and accumulative application completion time for Scenario 1. As the results show, node mobility has a significant impact on the performance of DRA and DICHOTOMY, whereas ADRA is less affected. In the case of two mobile nodes, the difference between the performance of ADRA and DRA is negligible. This is because both schemes allocated interdependent tasks to closely located nodes and the mobility of the two nodes did not have a significant effect on performance. However, as the number of mobile nodes increased, the performance of DRA was degraded by 70–180%

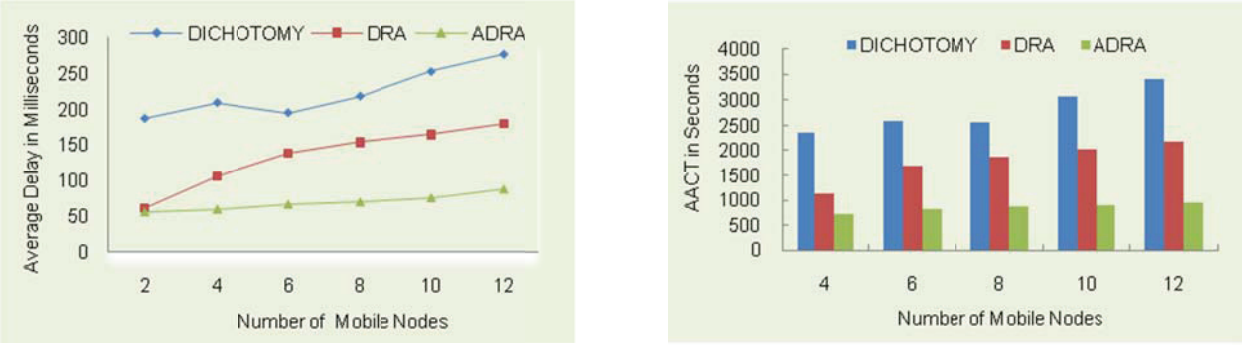

Figure 1. Average end-to-end communication delay and accumulative application completion time for Scenario 1

while the performance of ADRA was affected by only 20–60%. ADRA performed better due to an adaptation mechanism in which tasks were migrated to another node located nearby when a node executing a remote communication-bound task moved and crossed a distance threshold value. In the case of DRA and DICHOTOMY, it was observed that when a node executing a dependent task was moving away, an existing link between two nodes went down, which increased the control overhead. The increased communication distance also resulted in multi-hop communication, which significantly increased the overheads of the transport, routing, and medium access control layers in addition to the packet processing and queuing delays associated with application layer data at intermediate nodes.

Another trend that can be observed from the results shown in Figure 1 is that when the number of mobile nodes increases, the performance of DRA becomes stable. This is because most of the tasks executed on mobile nodes were either local communication-bound tasks or computation-bound tasks, which transfer small amounts of data and therefore have a minor effect on the performance. In the case of 4–6 mobile nodes, most of the tasks executed on mobile nodes were remote communication-bound tasks which exchanged large quantities of data and thus had an enormous effect on the performance.

Scenario 2: The ratio of remote communication-bound tasks to local communication-bound and computation-bound tasks is 2:3. The remaining setup is the same as that defined in Scenario 1.

The results for Scenario 2 presented in Figure 2 show that a small number of remote communication-bound tasks does not improve performance significantly. Both the DRA and the ADRA scheme have almost the same performance. This is because most of the tasks executed on mobile nodes were either local communication-bound tasks or computation-bound tasks, which are not considered in the adaptation mechanism. The ADRA scheme performs slightly better, because in some cases, particularly with 8–12 mobile nodes, nodes were executing remote communication-bound tasks which were migrated when the increased communication distance crossed the distance threshold value.

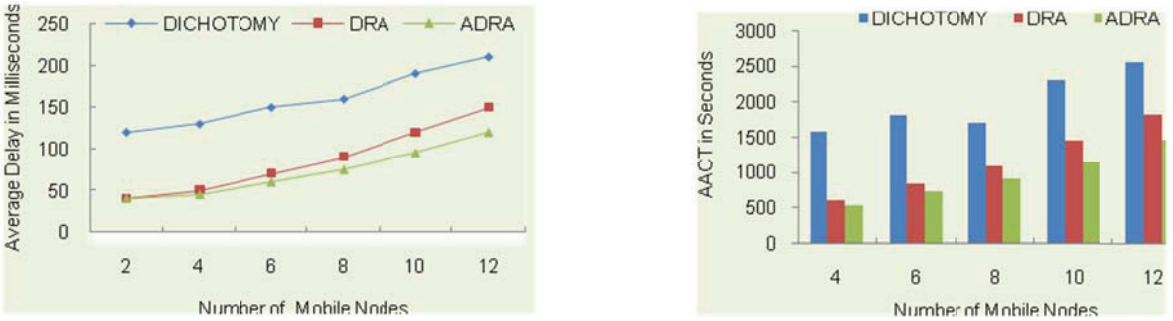

Figure 2. Average end-to-end communication delay and accumulative application completion time for Scenario 2

5.3.2. Average end-to-end communication delay with various numbers of tasks

Scenario 3: The ratio of stationary nodes to mobile nodes is 3:1. The remaining setup is the same as that defined above.

Figure 3 demonstrates the average end-to-end communication delay and accumulative application completion time for Scenario 3. As the results indicate, ADRA performs better and reduces the average communication delay by 40–80%. Compared to DRA, it improves performance by 50–60% with 12–24 tasks and by 40–50% with 36–48 tasks. Like ADRA, DRA initially allocated interdependent tasks to nodes located nearby, but after allocation, the mobility of nodes within a Grid increased the communication distance between interdependent tasks. The increased communication distance reduced path stability and also resulted in multi-hop communication, which significantly increased the communication cost. DRA also did not consider the task type, which further degraded the performance due to allocation of remote communication-bound tasks to nodes at multi-hop distances.

Compared to DICHOTOMY, ADRA improves performance by about 80% with 12–24 tasks and by about 55% with 36–48 tasks. DICHOTOMY allocates tasks based on processing power. It does not consider the distance between nodes or dependencies between tasks. Because high processing nodes were at multi-hop distances, the allocation of dependent tasks to high processing nodes resulted in multi-hop communication between dependent tasks, which increased the communication overhead.

Scenario 4: The ratio of remote communication-bound tasks to local communication-bound and computation-bound tasks is 2:3.

Scenario 4 reflects a small number of remote communication-bound tasks and thus a small amount of data transfers. As shown in Figure 4, with a small number of remote communication-bound tasks, ADRA does not achieve a significant performance gain with 12–24 tasks. However, as the number of tasks

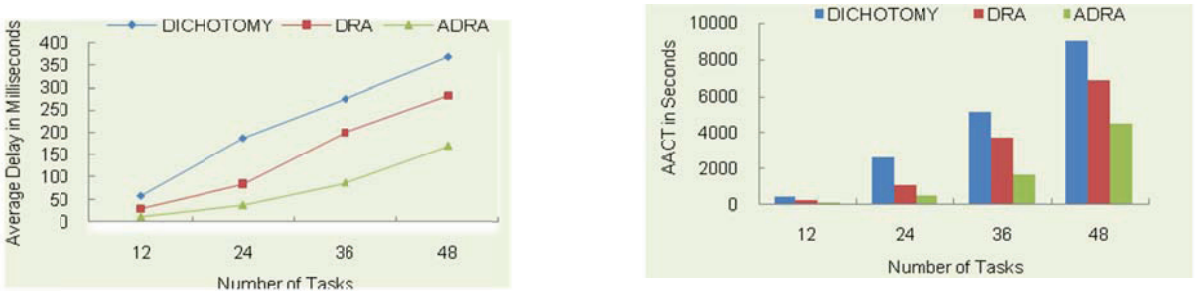

Figure 3. Average end-to-end communication delay and accumulative application completion time for Scenario 3

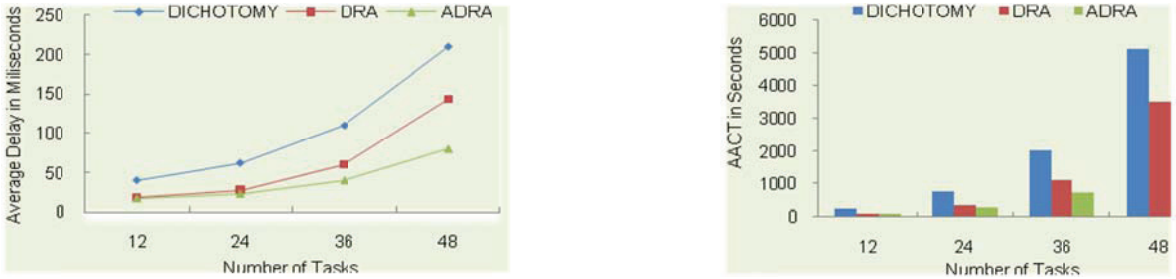

Figure 4. Average end-to-end communication delay and accumulative application completion time for Scenario 4

increases to 36–48, ADRA achieves significant performance gains. This is because with a small number of remote communication-bound tasks, most of the mobile nodes were executing either local communication-bound tasks or computation-bound tasks, which are not considered in the adaptation. However, as the number of tasks increased, the ratio of remote communication-bound tasks executed on mobile nodes also increased. Since remote communication-bound tasks are considered in the adaptation, the performance of ADRA became better.

A common trend that can be observed in Scenarios 3 and 4 is that the average end-to-end communication delay increases rapidly as the number of tasks increases. There are many possible reasons for this; two that we observed were a high level of network congestion where multiple remote communication-bound tasks were deployed and multiple access interferences. With an increasing number of remote communication-bound tasks, the communication traffic also increased, which resulted in network congestion. In addition, more tasks led to more competition to gain an access to the medium, which increased the delay. The routing overhead was another factor that increased significantly as the numbers of tasks increased from 24 to 48.

5.3.3. Accumulative application completion time

As mentioned earlier, two key components contribute to task completion time: execution cost and communication cost. Since communication cost is a product of the average end-to-end communication delay and the number of packets transmitted between tasks, a small improvement in average communication delay significantly reduces the task completion time and thus the accumulative application completion time. The results indicate that ADRA reduces the accumulative application completion time by 30–70%.

5.3.4. Energy consumption

The results for the energy consumption presented in Figures 5 and 6 indicate that ADRA significantly reduces energy consumption. In Scenario 1, it reduces energy consumption by 20–30% compared to DRA and by 30–40% compared to DICHOTOMY. In Scenario 2, however, it reduces energy consumption by 20–40% compared to DRA and by 30–50% compared to DICHOTOMY. This is because ADRA reduces the communication traffic generated by tasks and thus the energy consumed in the transmission of data.

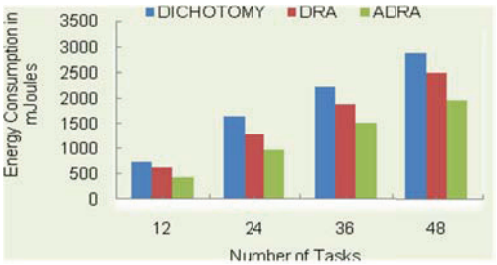

Figure 5. Energy consumption for Scenario 1

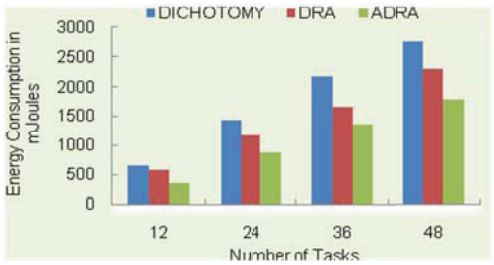

Figure 6. Energy consumption for Scenario 2

5.3.5. Comparison between centralized and fully decentralized architectures

The performance of ADRA based on centralized and fully decentralized architectures is demonstrated in Figure 7. As the results indicate, the ADRA based on a centralized architecture performs better. This is because it makes better allocation decisions due to the global view of the network. However, the ADRA based on a fully decentralized architecture results in ineffective allocation decisions due to the narrow view of the network. It also incurs an overhead because each node needs to exchange control information with neighboring nodes. However, this overhead contributes very little to the overall performance when a large amount of data transfer takes place between tasks.

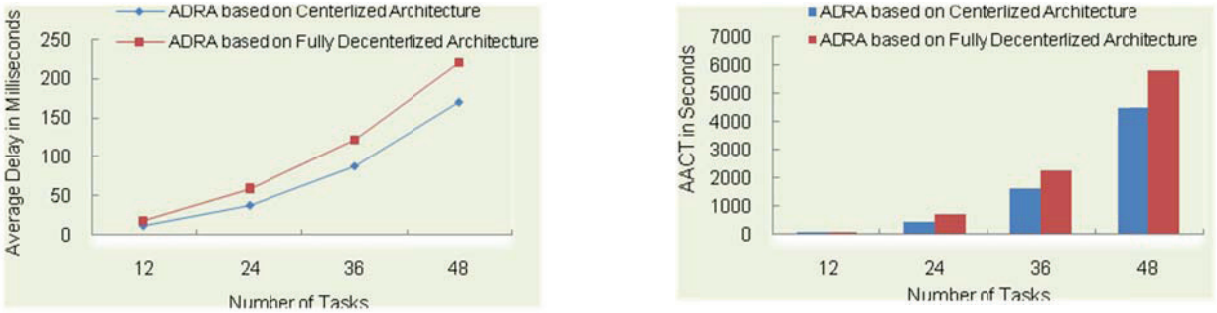

Figure 7. Performance of ADRA in terms of average end-to-end communication delay and accumulative application completion time using centralized and fully decentralized architectures

6. Conclusions

To reduce the cost of communication between dependent tasks, an adaptive and distance-based resource allocation scheme is proposed for allocation of interdependent tasks on a mobile ad hoc computing Grid. The scheme is based on a group mobility model and applies various heuristics to improve the performance of dependent tasks when nodes move within a Grid coverage area. The performance of the proposed scheme is compared with distance-based and DICHOTOMY resource allocation schemes through simulations. The results demonstrate significant performance gains.

Although the proposed scheme is focused only on mobile ad hoc computational Grids, we believe that it can be adopted in other domains such as multiple mobile robot systems, mobile Web services, mobile peer-to-peer computing systems, and so on where multiple nodes are involved to achieve a common goal or where a large amount of data transfer takes place between mobile nodes.

Acknowledgement

This research is a part of the project titled "WA-DGNSS Development" funded by Ministry of Land, Transport and Maritime Affairs, KOREA.

References

[1] **D. P. Agrawal, Q. A. Zeng.** Introduction to Wireless and Mobile Systems, *Thomson Brooks*, ISBN 0534-40851-6, 2003.

[2] **M. Arora, S. K. Das, R. Biswas.** A de-centralized scheduling and load balancing algorithm for heterogeneous grid environments, *in: Proc. Int. Conf. on Parallel Processing*, 18-21 Aug 2002, pp. 499-505.

[3] **M. Baker, R. Buyya, D. Laforenza.** Grids and Grid Technologies for Wire Area Distributed Computing Software — Practice and Experience, *John Wiley & Sons Ltd.*, 2002. 32(15):1437–1466.

[4] **R. K. Balan, M. Satyanarayanan, S. Y. Park, T. Okoshi.** Tactics-based remote execution for mobile computing. In: *Proc. MobiSys 2003, 1st Int. Conf. on Mobile Systems, Applications, and Services*, 5-8 May 2003, pp. 273-286.

[5] **T. D. Braun, H. J. Siegel, A. A. Maciejewski, Y. Hong.** Static resource allocation for heterogeneous computing environments with tasks having dependencies, priorities, deadlines and multiple versions, *J. Parallel Distrib. Comput*. 68 (2008) 1504–1516.

[6] **J. Cao, O. K. Kwong, X. Wang, W. Cai.** A peer-to-peer approach to task scheduling in computational grid, M. Li et al. (eds.), GCC 2003, *Lecture Notes in Computer Science* 3032 (2004) 316–323, http://dx.doi.org/10.1007/978-3-540-24679-4_65.

[7] **R.-S. Chang, J.-S. Chang, P.-S. Lin.** An ant algorithm for balanced job scheduling in grids, *J. Future Gener. Comput. Syst.* 25 (2009) 20–27, http://dx.doi.org/10.1016/j.future.2008.06.004.

[8] **B.-G. Chun, P. Maniatis.** Augmented smartphone applications through clone cloud execution (Intel Research Berkeley), *in: Proc. 12th Workshop on Hot Topics in Operating Systems*, 18-20 May 2009, pp. 8-9, http://dx.doi.org/10.1109/GRID.2004.44.

[9] **D. C. Chu, M. Humphrey.** Mobile OGSI.NET: Grid computing on mobile devices, *in: Proc. 5th IEEE/ACM Int. Workshop on Grid Computing*, 8 Nov 2004, pp. 182-191.

[10] **H. El-Rewini, H. H. Ali, T. Lewis.** Task scheduling in multiprocessing systems, *Computer* 28 (12) (1995), pp. 27-37, http://dx.doi.org/10.1109/2.476197.

[11] **D. G. Feitelson, L. Rudolph, U. Schwiegelshohn.** Parallel Job Scheduling – A Status Report, *Lecture*

Notes in Computer Science 3277 (2005) 1–16, http://dx.doi.org/10.1007/11407522_1.

[12] **G. Fox, A. Ho, R. W. Chu, E. I. Kwan.** A collaborative sensor grids framework. *IEEE Int. Symp. On Collaborative Technologies and Systems*, 18-22 May 2008, 29–38, http://dx.doi.org/10.1109/CTS.2008.4543909.

[13] **H. A. Franke, F. L. Koch.** Grid-M: Middleware to integrate mobile devices, sensors, and Grid computing. In: *Proc. 3rd Int. Conf. on Wireless and Mobile Communications,* 4-9 Mar 2007, pp. 12-20.

[14] **A. T. A. Gomes, A. Ziviani, L. S. Lima, M. Endler.** DICHOTOMY: a resource discovery and scheduling protocol for multi-hop ad hoc mobile grids. In: *7th IEEE Int. Symp. on Cluster Computing and the Grid*, 14-17 May 2007, 719-724.

[15] **P. Grabowski, B. L. Poznan.** Access from J2ME-Enabled Mobile Devices to Grid Services, *Supercomputing and Networking Center, Noskowskiego Poznan,* Poland, 2004, 61-704.

[16] **P. Ghosh, N. Roy, S. K. Das.** Mobility - aware efficient job scheduling in mobile grids. In: *7th IEEE Int. Symp. on Cluster Computing and the Grid,* 14-17 May 2007, 701-706.

[17] **F. Gonzalez-Castano, J. Vales-Alonso, M. Livny.** Condor grid computing from mobile handheld devices, *ACM SIGMOBILE Mobile Comput. Commun. Rev.* 6 (2) (2002), pp 117–126.

[18] **K. A. Hummel, G. Jelleschitz.** Robust de-centralized job scheduling approach for mobile peers in ad hoc grids. In: *7th IEEE Int. Symp. on Cluster Computing and the Grid*, 14-17 May 2007, pp. 461-470.

[19] **B. Hong, V. K. Prasanna.** Distributed adaptive task allocation in heterogeneous computing environments to maximize throughput. In: *Proc. 18th Int. Parallel and Distributed Processing Symposium*, 2004, p. 52, http://dx.doi.org/10.1109/IPDPS.2004.1302974.

[20] **C. Li, L. Li.** Utility-based scheduling for grid computing under constraints of energy budget and deadline. In: *Comput. Stand. Inter.* 31 (6) (2009), pp. 1131-1142, http://dx.doi.org/10.1016/j.csi.2008.12.004.

[21] **H. Liu, T. Roeder, K. Walsh, R. Barr, E. G. Sirer.** Design and implementation of a single system image operating system for ad hoc networks. In: *Proc. 3rd Int. Conference on Mobile Systems, Applications, and Services*, 6-8 June 2005, pp. 149–162, http://dx.doi.org/10.1145/1067170.1067187.

[22] **X. Liu, W. Wei, C. Qiao.** Task scheduling and lightpath establishment in optical grids. In: *IEEE INFOCOM*, 15-17 Apr 2008, 1966-1974.

[23] **D. Millard, A. Woukeu, F. B. Tao, H. Davis.** Experiences with writing grid clients for mobile devices. In: *1st Int. ELeGI Conf. on Advanced Technology for Enhanced Learning*, 14–16 Mar 2005.

[24] **X. H. Mario, G. C. Chuan.** A group mobility model for ad hoc wireless networks. In: *Proc. 2nd ACM Int. Conf. on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 1999, pp. 53–60.

[25] **J. M. Robinson, J. G. Frey, A. J. Stanford-Clark, A. D. Reynolds, B. V. Bedi.** Sensor networks and grid middleware for laboratory monitoring. In: *Proc. 1st Int. Conf. on e-Science and Grid Computing*, 5-8 Dec. 2005, pp. 562-569.

[26] **H. Rim, Y. Kim, H. Jo, S. Kim, H. Han, J. Kim.** Slim execution on distributed mobile environment. In: *Proc. 4th Int. Conf. on Asian Language Processing and Information Technology*, 2005, pp. 424–429.

[27] **A. C. Sodan.** A survey: loosely coordinated co-scheduling in the context of other approaches for dynamic job scheduling. In: *Concurr. Comp.-Pract*. E 17 (5) (2005) 1725-1781.

[28] **L. J. Senger, R. F. de Mello, M.J. Santana, R. H. C. Santana, L. T. Yang.** Improving scheduling decisions by using knowledge about parallel applications resource usage. In: *HPCC 2005, in: Lecture Notes in Computer Science*, 3726 (2005) 487–498, http://dx.doi.org/10.1007/11557654_57.

[29] **S. C. Shah, F. Hussain, M.-S. Park.** Resource allocation scheme to reduce communication cost in mobile ad hoc computational grids. *Int. J. of Space-Based and Situated Computing* 1 (4) (2011) 270–280.

[30] **S. C. Shah, M.-S. Park.** An energy-efficient resource allocation scheme for mobile ad hoc computational grids, *J. Grid Comput.* 9 (3) (2011) 303–323.

[31] **S. C. Shah, S. H. Chauhdary, A. K. Bashir, M. S. Park.** A centralized location-based job scheduling algorithm for inter-dependent jobs in mobile ad hoc computational grids, *Journal of Applied Sciences* 10 (3) (2010) 174–181.

[32] **S. Shilve, H. J. Siegel, A. A. Maciejewski, P. Sugavanam, T. Banka, R. Castain, K. Chindam, S. Dussinger, P. Pichumani, P. Satyasekaran, W. Saylor, D. Sendek, J. Sousa, J. Sridharan, J. Velazco.** Static allocation of resources to communicating subtasks in a heterogeneous ad hoc grid environment, *J. Parallel Distrib. Comput*. 66 (4) (2006), pp. 600–611.

[33] **V. V. Selvi, S. Sharfraz, R. Parthasarathi.** Mobile ad hoc grid using trace based mobility model, GPC 2007, *Lecture Notes in Computer Science* 4459 (2007) 274–285.

[34] **W. Su, S. J. Lee, M. Gerla.** Mobility prediction in wireless networks. In: *Proc. of IEEE MILCOM* 2000, pp. 491-495.

[35] **J. Yang, Y. Bai, Y. Qiu.** A decentralized resource allocation policy in minigrid. In: *Journal of Future Generation Computer Systems* 23 (2007), pp. 359–366.

[36] **J. Zhang, A. Hamalainen, J. Porras.** Addressing mobility issues in mobile environment. In: *Proc. 1st Workshop on Mobile Middleware: Embracing the Personal Communication Device*, Dec. 2008, pp. 1-5.

Received January 2011.