ISSN XXXX XXXX © 2017 IJESC

Research Article Volume 7 Issue No.6

Implementation of New Grid Computing Technology for Flexible Resource Sharing using MANET

Deekshith .K[1], Prithviraj[2]
Assistant Professor[1, 2]
Department of ISE
SDMIT, India

Abstract:
The Mobile Ad-hoc Network (MANET) opens a novel area for grid computing to extend pervasive computation in wireless environment. In this paper, a mobile agent based approach is proposed for building computation grid over MANET that is characterized as self-organizing and without fixed infrastructure. It distributes computations and aggregate resources through mobile agent. As an autonomous entity, it visits nodes with sufficient battery power and resource capability and negotiates their resource provision for running the computation job. It solves the problem of network topology change due to node mobility or power consumption, which may affect the stability of grid. Also it introduces flexibility in handling different cooperation models of nodes for resource sharing, thus is able to collect a lot of nodes in MANET to participate in grid computing. Mobile Ad Hoc Grid deals with the challenges in resource discovery and job scheduling due to its mobility and power consumption. In this paper a new grid computing technique is implemented to manage the congestion control with job scheduling. The proposed addition of the mobile ad hoc network within AHGL can offer maximum network utility.

Key words: Mobile Ad Hoc Networks, T-ALL Algorithm, Grid computing parameters, AHGL. Job scheduling in grid

I. INTRODUCTION

Not so long ago the target of the grid scientists in their projects and explorations were the wired networks. But recently there has been a fair amount of research on using computational grids over mobile environments. This is due to the expansion of mobile networks and the mobile technology, and the steady rise of their performances and computational capabilities. The potential ad hoc grid applications would allow nodes to accomplish specific task beyond an individual computing capacity. The ad-hoc networks are usually used for emergency applications when results are needed immediately. However, some calculations are often time consuming and need to be done in harshest environment. In order to obtain the results more quickly these types of applications can be deployed in an ad hoc grid. Some typical applications are in the field of disaster management, e-healthcare emergency, sharing resources, collaborative computing, etc. Current work concerning ad-hoc grid implementation can be divided into three groups. The first group deals with the challenges for establishing ad hoc mobile grids. The various challenges in ad-hoc grid like resource description, resource discovery, coordination systems, trust establishment and clearing mechanism are proposed. In different problems like finding the resources, scheduling the tasks, node discovery, node property assessment, service deployment, service security and the Quality of Service are analyzed. The second group deals with the challenges in resource discovery. The service advertisement method is clearly inadequate for dynamic resources such as CPU load, available memory, battery lifetime and available time of the nodes. A different solution for resource discovery is proposed using reactive protocols and it is characterized with the use of a new module for finding resources with the help of the existing AODV routing protocol. However, this approach shows the architecture only of the process responsible for finding the necessary services. The reactive routing protocols are finding that the nodes exchange messages between them about their resources and their neighbor's resources. The third group deals with task scheduling in the ad hoc grid environment. One proposed and existing solution that addresses this problem is to use the T-ALL algorithm. The main function of this algorithm is to allow resource provisioning to be scheduled among the most resourceful nodes in the mobile grid, while mitigating the overhead of discovery messages exchanged among them.

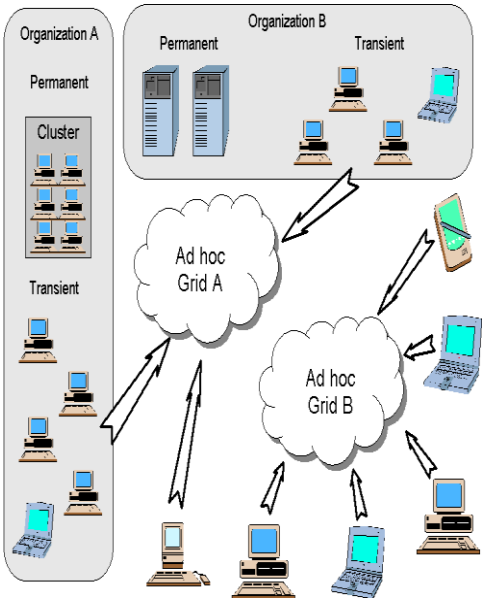

Figure.1. Ad hoc grid architecture overview

Grid computing in ad-hoc implementation needs to offer:
(1) Finding resources only when there is a need for execution of a given task.
(2) Minimum time of execution.
(3) No influence on the upper or the lower layers.
(4) Rescheduling if a certain node is down.
(5) Implementation on every platform.
(6) Data protection.
Also it must have the following characteristics:
(1) To keep pace with the dynamic nature of the network.
(2) Not to overload the network.
(3) To use different scheduling algorithms in order to improve the overall execution time.
(4) To intervene in case of node failure.
(5) To authenticate and authorize the nodes when joining the network.
Starting from the needs stated above we have created new implementation of Grid in ad hoc networks realized as a separate layer. This new independent Ad Hoc Grid Layer (AHGL) gives opportunity for scalability without any affection on the other layers of the network architecture. The AHGL reduces the unnecessary traffic in the ad hoc grid environment because it includes only the nodes that satisfy the criteria for execution of certain jobs.

II. AD HOC ON-DEMAND DISTANCE VECTOR ROUTING (AODV) PROTOCOL

The Ad hoc On-demand Distance Vector (AODV) routing protocol is a reactive MANET routing protocol. The AODV broadcasts a route request to discover a route in a reactive mode. In AODV a field of the number of hops is used in the route record, instead of a list of intermediate router addresses. Each intermediate router sets up a temporary reverse link in the process of a route discovery. This link points to the router that forwarded the request. Hence, the reply message can find its way back to the initiator when a route is discovered. When intermediate routers receive the reply, they can also set up corresponding forward routing entries. To prevent old routing information being used as a reply to the latest request, a destination sequence number is used in the route discovery packet and the route reply packet. A higher sequence number implies a more recent route request. One advantage of AODV is that AODV is loop-free due to the destination sequence numbers associated with routes. The Bellman-Ford algorithm avoids "count to infinity" problem. Therefore, it offers quick convergence when the ad hoc network topology changes, which typically, occurs when a node moves in the network. Poor scalability is a disadvantage of AODV. We use the example topology shown in Figure 2 to illustrate the discovery procedure of AODV. Note that Routers *A* and *C* are disconnected from each other while both of them connect to *B*. When Router *A* starts a route discovery to *C*, a route request is broadcast. The request packet contains the requested destination sequence number, which is 1 greater than the one currently kept at *A*. For example, assume that the destination sequence number for *C* at *A* is 0x00000000, then the destination sequence number in the 20 route discovery packet is 0x00000001. The intermediate routers reply to the source if they know the route to that destination with the same or higher destination sequence number. We assume that *B* does not have a record for a route to *C*. Therefore, *B* first sets up a temporary link pointing back to *A*. In the second step, it increases the number of hops by 1 and rebroadcasts the request. When *C* receives that request, it creates a new destination sequence number. A route reply with that new sequence number is sent by *C*. The initiator and all intermediate routers build routing entries associated with this new sequence number when they receive the reply. The number of hop values can be used to find a shorter path if a router receives two replies with the same destination sequence number. AODV also handles unreliable transmission of control messages.

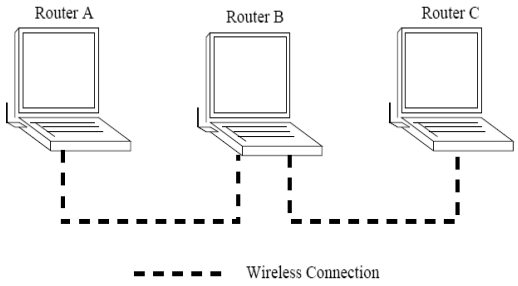

Figure.2. Example of AODV routing protocol

III. GRID PROTOCOL ARCHITECTURE

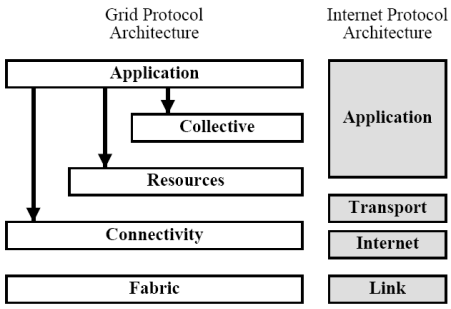

Figure 3. The layered Grid architecture and its relationship to the Internet protocol architecture.

Grids provide protocols and services at five different layers as identified in the Grid protocol architecture (see Figure 3). At the **fabric layer**, Grids provide access to different resource types such as compute, storage and network resource, code repository, etc. Grids usually rely on existing fabric components, for instance, local resource managers and general-purpose components such as GARA (general architecture for advanced reservation) and specialized resource management services such as Falkon(it also provides services beyond the fabric layer). The **connectivity layer** defines core communication and authentication protocols for easy and secure network transactions. The GSI (Grid Security Infrastructure) protocol underlies every Grid transaction. The **resource layer** defines protocols for the publication, discovery, negotiation, monitoring, accounting and payment of sharing operations on individual resources. The GRAM (Grid Resource Access and Management) protocol is used for allocation of computational resources and for monitoring and control of computation on those resources, and GridFTP for data access and high-speed data transfer. GridFTP is a secure and reliable data transfer protocol providing high performance and optimized for wide-area networks that have high bandwidth. As one might guess from its name, it is based upon the Internet FTP protocol and includes extensions

that make it a desirable tool in a grid environment. The GridFTP uses basic Grid security on both control (command) and data channels. Features include multiple data channels for parallel transfers, partial file transfers, third-party transfers, and more. GridFTP can be used to move files (especially large files) across a network efficiently and reliably. These files may include the executables required for an application or data to be consumed or returned by an application. Higher level services, such as data replication services, could be built on top of GridFTP. The **collective layer** captures interactions across collections of resources, directory services such as MDS (Monitoring and Discovery Service) allows for the monitoring and discovery of VO resources, Condor-G and Nimrod-G are examples of co-allocating, scheduling and brokering services, and MPICH for Grid enabled programming systems, and CAS (community authorization service) for global resource policies. The **application layer** comprises whatever user applications built on top of the above protocols and APIs and operate in VO environments. Two examples are Grid workflow systems, and Grid portals.

IV. MOBILE AD HOC NETWORK

A mobile ad hoc network is formed by mobile hosts. Some of these mobile hosts are willing to forward packets for neighbours. MANET is characterized as an infrastructure-less mobile wireless network, in which two mobile nodes communicate with each other through intermediate nodes. Since there is no explicit server, every mobile node should work autonomously. In an ad hoc network, the topology changes very frequently, as nodes may join and leave the network. Thus routing in ad hoc networks has to be different from routing in wired networks to handle the issue of mobility, frequently changing topology, route breaks, low band-width and high delay, interference, limited energy of mobile nodes etc. A variety of routing protocols for multi-hop ad hoc networks have been proposed.

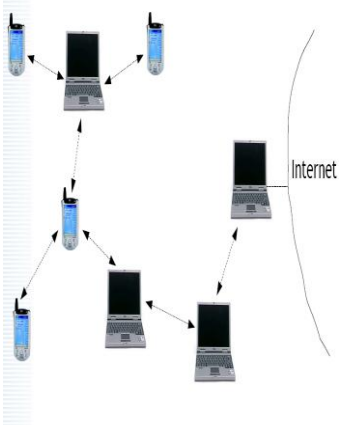

Figure.4. Example of a MANET

V. AD HOC GRID LAYER (AHGL)

The Ad Hoc Grid Layer is designed as a separate layer in the network architecture of all the nodes in the ad hoc grid environment (fig. 5). Depending on the function of the node the layer can be in Master or Slave mode. AHGL for a given node is in master mode when that node is supposed to find free resources throughout the ad hoc network, to schedule jobs and to dispatch them to the rest of the nodes. The AHGL for nodes that are meant to receive jobs, execute them and reply, with the results is in slave mode.

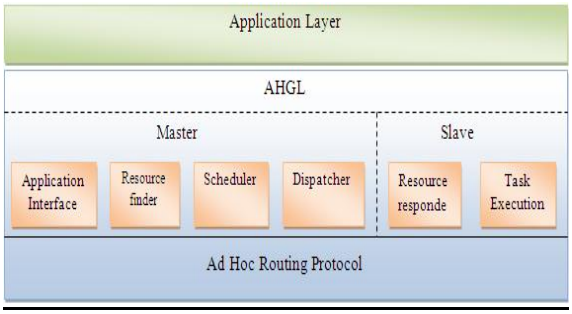

Figure.5. Position of the new AHGL in the ad hoc grid network architecture

All of these layer functions are realized by the means of different services. Depending on the role of the node, the particular services are activated. In AHGL there are implemented six possible services:

(1) Application interface service
(2) Resource finder
(3) Scheduler
(4) Dispatcher
(5) Resource response service
(6) Job execution service

APPLICATION LAYER
AHGL

(A) Application interface service

The first goal of the AHGL layer is to divide the application on small grid jobs with structure as shown in table 1. This is done by the application interface service. Taking into consideration that every node may have several applications to execute, their distinction is done by the field ID_{app}. Also jobs which are part of the same application must be distinguished. This is done using the ID_{job} field. The last field ID_{SW} is used to specify the software that is needed for proper execution of the job. The data needed to execute a certain job is in the DATA field.

Table. 1. Representation of a job

ID_{app}	ID_{job}	ID_{SW}	DATA

According to the specified software (ID_{SW}) the Application interface service calculates the size and the number of instructions that every job has. These parameters are used to check if the job can be placed in the available memory of the node and to calculate the time needed for job execution. The output for this service is in form of a table of parameters (table 2). The first and the second column (ID_{app} and ID_{job}) are used for identification of the jobs; the third column is used for the length

of the job in bytes and the fourth is the number of instructions (N_{ins}).

Table .2.Example of the parameters table

ID_{app}	ID_{job}	L_{job}	N_{job}
1	12	4378	212
1	3	6723	329

For example, the first row presents the parameters of a job that belongs to the first application, the job has ID=12, size of 4378 KB and 212 instructions.

(B) Resource Finder

The dynamic properties of ad hoc networks do not allow a central register node. This is the reason why we have implemented a service which discovers free resources across the network in the moment when there are incoming jobs to be executed. This is done by the resource finder service that generates AHGL request packet and broadcasts it to all available nodes in the network. The structure of AHGL request packet is shown in table 3.

Table. 3.AHGL request packet

ID_{node}	SEQ	CPU	MEM	BAT

The first field is the destination of the packet (ID_{node}) while the second field (SEQ) is used to distinguish the different requests from a same node and to escape forever loops. To reduce traffic and to exclude the nodes that cannot execute even the smallest jobs, we are using the minimalism criterion. The following fields are parameters that every node must satisfy in order to be considered as a part of this ad hoc grid environment. These parameters are the minimum speed of the processor in GHz (CPU), the minimum free memory in KB (MEM) and minimum amount of the battery (BAT). Nodes that satisfy the given criterion reply with an AHGL reply packet shown on table 4. Using this packet, the nodes announce their processor speed (CPU), amount of free memory (MEM) and battery status (BAT). For future calculation of time needed to transmit the packet we also count the number of the hops needed to reach node (H). Nodes that are offering free resources are available for certain period of time. This free period of time starts at T_S and ends at T_E. During this period, nodes can be used in a given effective interval (T_{EFF}).

Table .4.AHGL reply packet

ID_{node}	SEQ	CPU	MEM	BAT	H	T_S	T_E	T_{EFF}

The resource finder service creates the resource table using the received AHGL reply packets, which is presented in table 5, and afterwards used by the next service (scheduler) as input for the execution time optimization.

Table. 5.Example of resource table

ID_{node}	CPU	MEM	H	T_S	T_E	T_{EFF}	RTT
5	2.5	284850	2	935	8625	2500	1.7452
27	2.0	120057	1	1101	1566	2	0.4527

For example, the first row presents the information obtained from node with ID=5 that has processor speed of 2.5 GHz, free memory of 284850 KB, is reachable in two hops from the master. It is available in time interval between 935 s and 8625 s, offers effective interval of 2500 s, and has Round Trip Time (RTT) of 1.7452 seconds.

(C) Scheduler

The purpose of this service is to find the optimum time for execution of all jobs using a given scheduling algorithm, where the process is influenced by different parameters. Such parameters are: start and end time of the nodes availability (T_S, T_E), effective time (T_{EFF}), processor speed (CPU), number of instructions (N_{ins}), job size, network throughput (R), number of hops, round trip time (RTT) and the job priority. All those parameters are read from resource table and parameters table. The scheduling process first checks the time of the availability of the node (T_{EFF}). It also checks whether the node offers enough time for execution of the job and whether the free memory of the node is big enough for the size of a certain job. If these criteria are achieved, the job is assigned to that node and its effective time is updated. The result of algorithm is a dispatch table 6 announcing which job is to be executed at which node.

Table. 6. Example of dispatch table

ID_{node}	ID_{app}	ID_{job}
12	1	14
20	1	3

For example, the first row represents a job with ID=14 that belongs to the first application which is assigned to the node 12 for execution.

(D) Dispatcher

The Dispatcher is the fourth service in AHGL and its job is to dispatch the jobs to the appropriate nodes and to forward the obtained results to the Application interface service. This service uses the dispatch table, parameters table, and the queue with the jobs and it creates the job packet (table 7). The parameter table is used by the dispatcher to send jobs to the assigned node until the node has a free amount of memory. The packets are transferred to the lower layers and using the ad hoc routing protocol, sent to the destination nodes.

Table.7. Job packet

ID_{node}	ID_{app}	ID_{job}	ID_{SW}	DATA

The first field ID_{node} is used as destination for job packet, while the other fields are used to represent the job (ID_{app}, ID_{job}, ID_{SW} and DATA). After the execution of the job, the node generates a result packet.

Table .8.Result packet

ID_{node}	ID_{app}	ID_{job}	RESULT

Because of the nature of the ad hoc networks, there is a great possibility that a certain node will not send a result as a consequence of a node failure or connection failure. In this case, the jobs assigned to that node, must be rescheduled. Scheduling

based on task transmission and execution time (T-ALL) is the last new proposed scheduling algorithm. This algorithm is based on transmission and execution time of each task on every node that is part of Grid ad hoc environment.

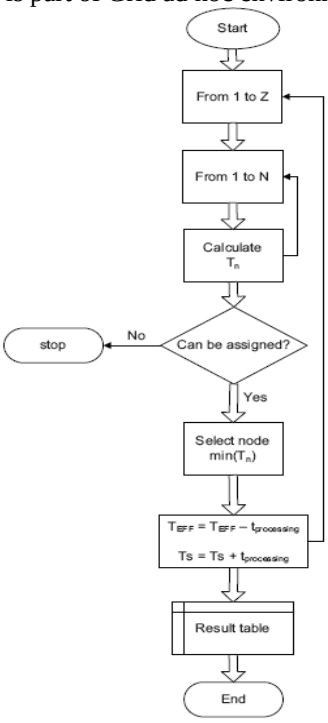

Figure. 6. T-ALL scheduling algorithm

Figure 6 present the flow of this algorithm. Z denotes number of all tasks and N =number of nodes. Tasks are selected in the same order as they arrive in the queue. With Tn we denote time that is needed to send the a task at node n, execute, and return the result. Algorithm has to pass and calculate tasks transmission and execution time for all nodes (n=1...N), and as a result to assign the task to the node which will return result for minimal time. If the nodes have insufficient resources to execute the tasks then all process of scheduling is stopped. During the process of task scheduling, T-ALL algorithm also takes into consideration the time when node start to be available (T_S), and the effective time of the node (TEFF), by updating this parameters when a task is assigned to the node. As the end, T-ALL algorithm will provide a table with information about which task on which node is assigned.

(E) Resource Response Service

This service monitors the available resources for a given node and it responds when AHGL request packets are received. First it checks the sequence number (SEQ) of the request then it compares the fields: processor (CPU), free memory (MEM) and battery energy (BAT) from the packet with the actual values from the node. If the values from the node are bigger, then the criteria are satisfied and the node generates AHGL response packet (table 4), which is then transferred to the lower ad hoc routing layer.

(F) Job Execution Service

Job execution is the last service and is activated when the job arrives at the slave node. The job is forwarded to the upper layer for execution and the obtained results are packed and sent to the master node.

VI. CHALLENGES

The main steps which need to be taken to allow ad hoc grid computing in a heterogeneous environment are as follows.

- **Node Discovery:** In an ad hoc grid environment, the network topology is dynamic (i.e. rebooting of workstations, movement of laptops, replacement of computers) and thus a node detection solution geared towards frequent node arrivals and departures is required. While arrival or departure of nodes should be discovered as quickly as possible, a balance needs to be found between keeping the topology information up to date and flooding the network with discovery messages.

- **Node Property Assessment**

OGSA and OGSI define virtualization of available resources at the system-independent level of resource access to allow uniform access to a heterogeneous system. The hardware and operating system or underlying implementation of a service is hidden from the caller of a service. Support of the standard grid service interfaces is guaranteed and sufficient to allow access to the resource. Even the instantiation of a deployed grid service is system independent, since it is specified to be handled by a gate keeper (service factory).To enable autonomous deployment, meta-information from the grid participant must be available to the deployment service, so it can reliably operate in a heterogeneous environment. While the underlying system of each node is guaranteed to support the grid service interface, the way it implements this is not specified by OGSA or OGSI. Furthermore, information on the reliability of the nodes can be taken into account when services are to be deployed, so nodes with long up times are given priority over nodes which frequently reboot or crash.

- **Service Deployment**

Vital to the invocation of services on various nodes in the grid system is the availability of those services. For a large scale ad hoc grid, the time consumption for manual deployment is prohibitive, and management is difficult due to the fluctuating availability of the nodes. Even with the availability of advanced grid programming toolkits deployment of services has been identified as a critical issue. Furthermore, the number of grid services will steadily rise with the number of users, further increasing the management cost of the grid environment. In a dynamically changing environment, deployment is even more critical as there is no single deployment cycle that reaches all machines. Instead, services need to be deployed and instantiated on demand on machines as they become available. Service deployment becomes part of an ad hoc grid application instead of being handled by a system administrator as a precondition to the use of a service. In a production environment, every operation needs to be non-intrusive, i.e. it does not interfere with the execution of other services already running on the grid.

- **Service Security**

Security is a major aspect in all distributed systems, since there is always the possibility that a node introduces malicious code. In an ad hoc grid, several new aspects must be dealt with beyond the standard security requirements existing in previous grid systems. In traditional systems, installing a service requires security certificates allowing the operations. Services usually can

only be installed by a very small number of people and trust can be assumed between all parties. In a large ad hoc system on the other hand, it is possible that users unknown to each other operate on the same node. This gives rise to new security issues. One major new security threat is that a trusted node is running further unknown services. Here, inter service security must be offered, since fair play is not guaranteed any more.

VII. CONCLUSION

In this paper, we presented a new layer for grid computing in ad hoc networks called Ad Hoc Grid Layer (AHGL).This layer is a result of the implementation of the grid services in an ad hoc network. Its position in the network architecture allows this layer to be independent from the upper or the lower layers and it allows new services to be added without changing the other layers. AHGL was created in a manner that corresponds to the ad hoc network specific features. It is designed to be able to adapt to the network dynamical environment with a high fault tolerance degree. The implementation allows us to investigate the advantages and the possible weaknesses of the model. It also gives the opportunities to make an optimization of the different parameters that influences the performances of the ad hoc grid. The scheduler service of AHGL offers large number of different kind of parameters for the network and nodes that can offer implementation of a different scheduling algorithm.

VIII. REFERENCES

[1]. Aksenti Granarov, Bekim Cilku, Igor Miskovski, Sonja Filiposka and Dimitar Trajanov, "Grid Computing Implementation in Ad Hoc Networks", Advances inComputer and Information Sciences and Engineering,Springer, 196-201, 2008.

[2]. Heba Abd El-Rahman, Amal El-Nahas, Mohamed Hashem, "LSLP: A Location Based Service Location Protocol for Service Discovery in Mobile Ad Hoc Networks", Fourth International Conference onIntelligent Computing and Information Systems, pp:791-798,2009.

[3]. Imran Ihsan, Muhammad Abdul Qadir and NadeemIfikhar, "Mobile Ad Hoc Service Grid – MASGRID",PWASET, Vol 5, 2005, ISSN 1307-6884.

[4]. Zhoqun Li, Lingfen Sun, Emmanuel C, Ifeachor, "Challenges of Mobile Ad Hoc Grids and their Applications in E-Health Care". in the Proceedings of Second International Conference on Computational Intelligence in Medicine and Healthcare (CIMED 2005)-2005.

[5]. R. Baldoni, R. Beraldi, and A. Mian. Survey of service discovery protocols in mobile ad hoc networks. Technical Report MidLabDip. Informatica e Sistemistica, Universita di Roma, 2006.

[6]. Zhi Wang, Bo Yu, Qi Chen, Chunshan Gao. Wireless Grid Computing over mobile ad- hoc networks with mobile agent. 0-7695-2534-2/05 IEEE 2006.