Performance Analysis of Congestion Control in Mobile Adhoc Grid Layer

R.Bhaskaran and M.Madheswaran
Centre for Advanced Research
Muthayammal Engineering College Rasipuram, Tamilnadu

ABSTRACT
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 with in AHGL can offer maximum network utility. The model is implemented in the NS-2 network simulator.

Categories and Subject Descriptors
C.2.1- Advantage networks (Network Simulator)

General Terms
Algorithms, Performance, Design.

Keywords
Mobile Ad Hoc Grid, Congestion Control, AHGL, GPD

1. INTRODUCTION
Advances in ubiquitous computing and mobile networking technologies have created a new challenge for implantation of Grid computing in ad hoc network environments. The resource sharing prospective of Grid computing opens up the amalgamation of Grid technology with mobile ad hoc networks. The integration of the resource aggregation model of Grid with mobile ad hoc platform that can be instantly constructed anytime and anywhere. The mobile ad hoc grid technology can be applied for disaster management, wild fire fighting, e-health care and emergency. [1]

The general attributes of mobile ad hoc networks are bandwidth, power, Multi-hop delivery, Network partitioning and Infrastructure unpredictability. The challenges in mobile ad hoc grid are energy consumption, network topology which is neither stable nor predictable, bandwidth – limited and vulnerable to adverse signal quality, without a reliable central infrastructure, the organization of grid resources and execution of grid applications require decentralized operation, fault tolerance and more difficult to perform authentication and to provide general security mechanisms. The two main functionalities of the mobile ad hoc grid are service discovery, job scheduling [6].The concept of resource sharing in grid computing is realized by service discovery mechanisms [2][3]. It locates available resources/services upon request throughout the grid infrastructure. The job scheduling in grid computing is to allocate resources to jobs for a certain amount of time.

In this paper, the general challenges in implementing job scheduling in the mobile ad hoc environment and the implementation of congestion control in job scheduling are analyzed. This implementation can speed up the data flow in mobile ad hoc network.

2. JOB SCHEDULING IN MOBILE ADHOC GRID
Job scheduling in Mobile Ad Hoc Grid is to allocate resources to jobs. The scheduling algorithm [7] depends on the requests of the application, which ranges from processing node, data to network bandwidth, signal filters, etc., Requests are defined as jobs, which consists of information about the requirement on the resource that are necessary to execute a particular job. [5]The system architecture of job scheduling can be categorized as centralized scheduling, hierarchical scheduling and decentralized scheduling. In centralized scheduling, central scheduler collects the information of network resources. This is not scalable well with increasing size of the grids and become a bottleneck in case of node failure. In Hierarchical a scheduling different policy are used for local and global scheduling, and not depends on the single scheduler. In decentralized scheduling the distributed schedulers interact with each other and schedule jobs at remote nodes for execution. Local schedulers can submit jobs to/from each other through direct communication or a central job pool. This type of job scheduling is better for fault tolerance and reliability. This proposed work is based on the decentralized on – demand job scheduling. This can meet the challenges such as alert for node mobility, robust in network partitioning and fault tolerant upon data loss in mobile ad hoc grids.

3. CONGESTION CONTROL IN MOBILE AD HOC GRID
[4]Mobile ad hoc scheduling algorithms is responsible for congestion control as well as the job scheduling process. The congestion control decides which packets to be injected into the flows in each time. The next packet to be transmitted is determined by the job-scheduling algorithm. Job scheduling with congestion control is based on primal dual algorithm solves NUM (Network Utility Maximization) problem.[4]

$$\max \sum_{f} U_f\left(x_f\right) \tag{1}$$

Subject to

$$\sum_{f \in s_e} x_f \le c_e \tag{2}$$

where C_e be the capacity of the server, S_e be the set of flows passing through the server and X_f be the current injection rate into flow *f*. $U_f(.)$ is defined as the utility function that represents the "benefit to the system" achieved by the given flow rate

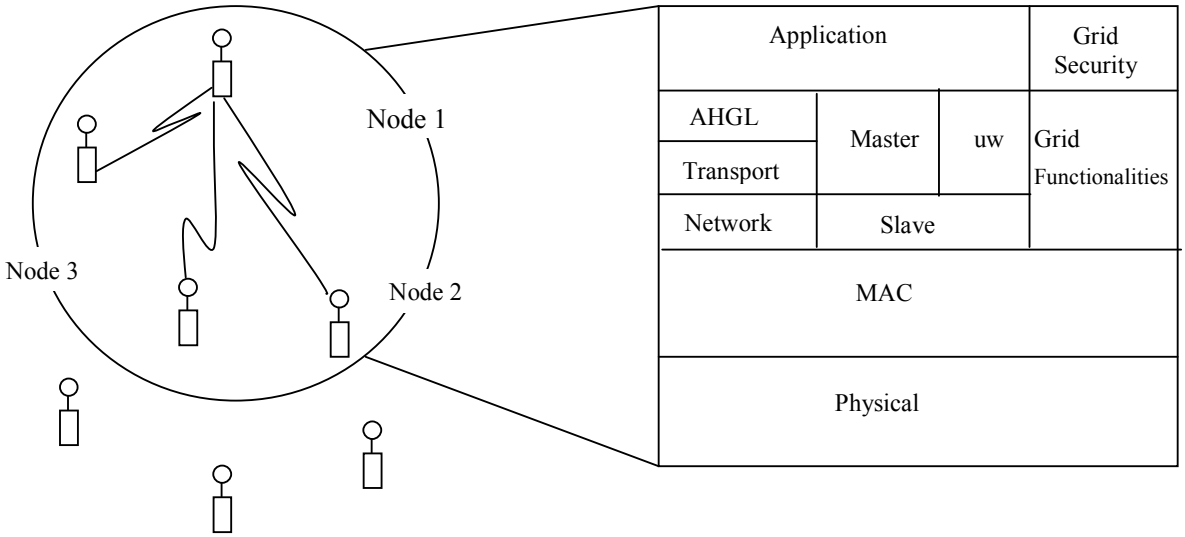

Fig 1. Architecture of modified M-AHGL

The network is considered to be under loaded, as long as all of the capacity constraints are satisfied. This NUM problem is achieved by Greedy Primal Dual (GPD) algorithm. The nodes with in the schedulable region is defined by the ordered tuples (i,j,r_{ij}), where i is the sender, j is the receiver and r_{ij} is the injection rate. The capacity region defines flow rate only for the nodes in schedulable region.

The congestion control and scheduling is performed by the following steps:

1. PDQ maintenance
2. Scheduling and contention resolution
3. Signaling
4. Congestion control

According to GPD each node maintains one Per Destination Queue (PDQ) denoted by Q_d^i. Which contains all the packets at node i that have address d as their destination. The amount of data in PDQ is defined by q_d^i. The next hop for destination d bounds traffic after it leave node i, that is defined as $n(i,d)$. Each queue has an associated urgency weight w_d^i. Urgency weights are the inputs to the scheduling component of GPD, which is calculated by

$$\mathrm{w}_{\mathrm{d}}^{\mathrm{i}} = \left[q_d^i - q_d^{n(i,d)}\right] r_{i,n(i,d)} \quad (3)$$

where $n(i,d)$ is the next hop after node i in the route to destination d, $r_i n(i,d)$ is the channel rate between node i and node $n(i,d)$.

In scheduling and contention resolution, the scheduling decisions have two processes such as intra node scheduling and inter node scheduling (contention resolution scheduling). In intra node scheduling each node decides the transmission of packets. Contention resolution scheduling decides the transmission of the next node. The PDQ is decided by the node whenever the transmission is accomplished whose is maximum. It then removes a packet from the PDQ and joins the contention resolution contest to conclude among the contending set of transmissions to allow transmitting.

The overall contention is reduced by allowing the nodes with highest urgency weight for channel access for which inter node scheduling are done. It replaces the urgency weight w_d^i defined in GPD with $f\left(w_d^i\right)$ where $f(.)$ is a "steeply" increasing function. Signaling enhances the carrier sensing mechanism in which node can hear interference at a range greater than their transmission range. Congestion control can be applied for both reliable version and unreliable version of data flow. In unreliable version of the congestion control protocol, and decision is made whether or not to inject a packet whenever it arrives from the application layer. The decision is made by every flow f maintain an average rate x_f that is exponentially filtered average of the amount of data admitted to the flow. The time constant for this filter is small parameter β. x_f is multiplied by a factor $1-\beta$ in each time step and is increased by βl_p whenever a packer of size l_p is injected into flow f.

For elastic flows a packet is injected as long as $U`\left(x_f\right) - \beta q_{d_f}^{s_f} > 0$, where $U`_f(\bullet)$ is the first derivative of $U_f(\bullet)$. For semi elastic a packer is injected as long as $g\left(\beta k_f\right) + U`\left(x_f\right) - \beta q_{d_f}^{s_f} > 0$, where $g(.)$ is some increasing function and k_f is a token counter that keeps track of whether or not flow f is meeting its minimum rate

requirement. k_f receives tokens at rate $R_f^{\min}$ at all times. Whenever a packet of size l_p is injected into flow f, k_f is decremented by an amount l_p. The reliable version dataflow of the protocol needs sequence numbers

and acknowledgements to keep track of received bytes at the destination.

4. IMPLEMENTATION OF CONGESTION CONTROL IN AHGL

This proposed work integrates both the congestion control and job scheduling in the AHGL of mobile ad hoc grid. The nodes in master mode of AHGL are responsible for scheduling and congestion control. The application interface service of AHGL divides the application into small grid jobs. The job format is shown in Fig., 2.

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

Figure 2. Format of a small grid job

Where ID_{app} is an application identifier used to distinguish the jobs, which are part of the same application. ID_{job} is the job identifier, ID_{sw} is the type of software needed to execute the job, U_w is the urgency weight initialized as 0, it can be varied with the congestion control process on entering the PDQ and *DATA* is data to be processed by the job. [1]The dynamic topology and mobility of mobile ad hoc grid not support the central register node to maintain the resource table. The free resources are discovered with On-demand resource discovery applied to AHGL at the moment jobs enter the mobile ad hoc grid, generates AHGL request packet and broadcast it to all available nodes, which is shown in fig., 3.

ID_{node}	SEQ	CPU	MEM	BAT

Figure 3. Structure of Modified AHGL request packet

The minimum requirements to process the job are defined in the AHGL request packet. ID_{node} is the destination of the packet, *SEQ* used to distinguish different requests from the same node, *CPU* defines the required minimum speed of the processor in GHz, *MEM* defines the required minimum free memory and *BAT* is the required minimum amount of battery. Nodes that satisfy the criterion reply with an AHGL reply packet of the given structure in fig. 4. These nodes offer free resources for certain period of time. Resource finder creates the resource table using the received AHGL reply packets.

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

Figure 4. Modified AHGL Reply Packet

The parameters passed by the node in reply packet defined as *CPU is* the actual processing speed, *MEM* is the available amount of free memory, *BAT* is the current battery status, *H* - the number of hops needed to reach the node T_s defines startup of free time. T_E defines end of free time and the effective interval is defined by T_{EFF}. This scheduler integrates the scheduling process and congestion control based on the Greedy primal dual algorithm to solve NUM problem defined in equation (1) and (2).

For every inter node scheduling and contention resolution is performed by using urgency weight. Signaling is performed to sense the interference at a range greater than their transmission range. The congestion control is performed on both reliable version and unreliable version of data
flow to inject a packet that can obtain the utility function needed for each traffic flow. The job sequence which can satisfy the NUM [4] problem is constructed by the joint scheduling and congestion control are dispatched to the appropriate nodes. Application interface service uses dispatch table, parameter table. and jobs in queue. Dispatcher generates the job packets in the format shown in Fig. 2. The resource response service monitors the available resources for a given node and responds to each AHGL request packet. It checks the sequence number *SEQ* of the request and compares the fields *CPU, MEM, BAT* from the request packet with the actual values from the node, if the values from the node are higher than criteria it satisfied AHGL response packet is generated and transferred to ad hoc network routing layer. The job execution service is activated when the jobs arrive at the slave node. The jobs are forwarded to the upper layer for execution and obtained results are transferred to the master mode

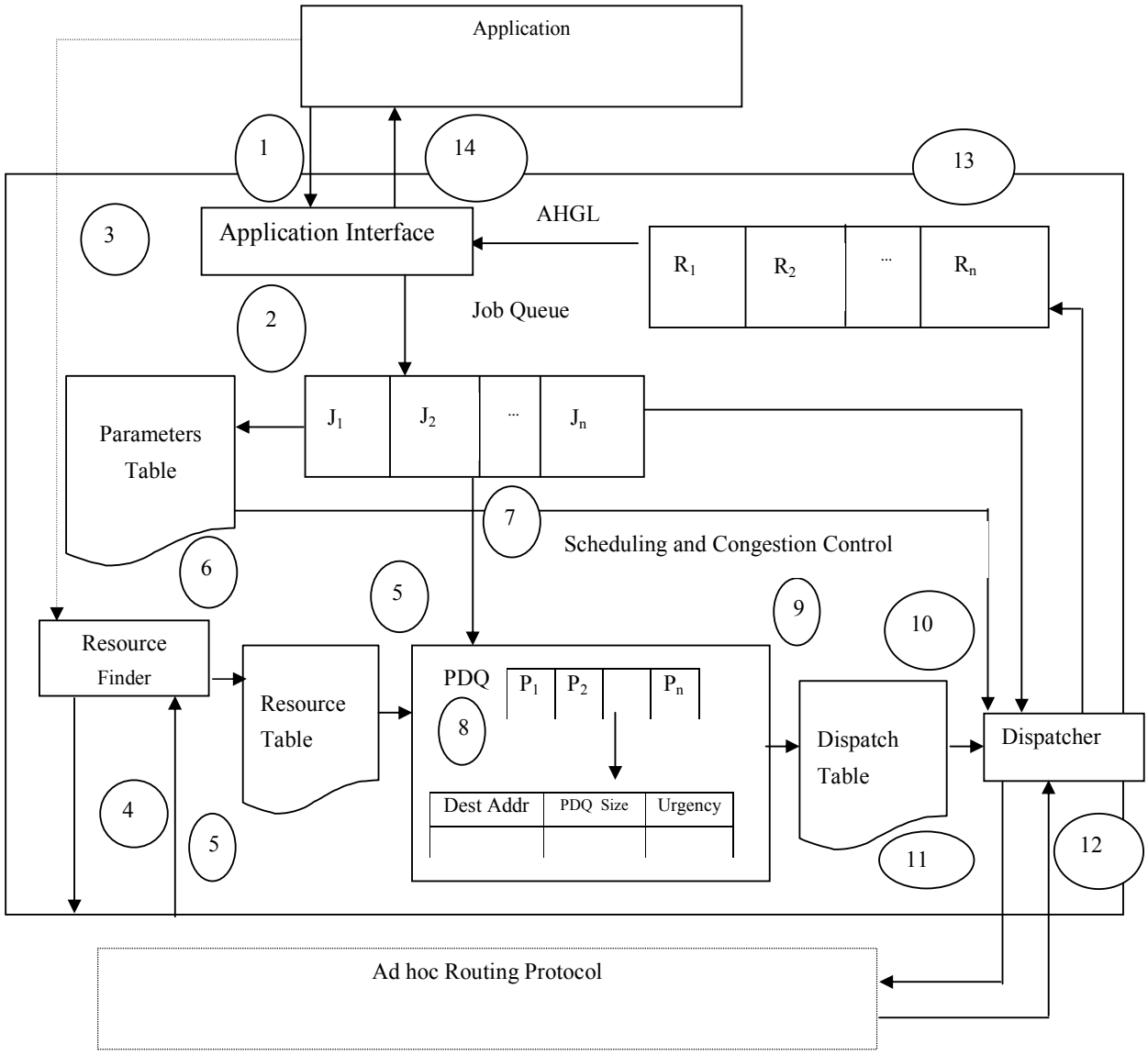

Fig 5. Architecture of Modified m-AHGL

5. Modified AHGL in Master Mode / Slave Mode

Application interface in AHGL master mode receives the application and separate them into smaller grid jobs, which can form a job queue. It calculates the parameter for every job and constructs the parameter table. The resource finder service is activated and it issues the AHGL request packet. The nodes which receive the packet send the AHGL reply packet. The resource table is constructed by collecting all the replies from the slave nodes. The joint scheduling and congestion control is responsible for inter node and intra node job scheduling, calculates the urgency weight for PDQ's and perform congestion control to minimize the network traffic. The congestion control information is transferred to the transport layer. The Dispatcher constructs the dispatch table and dispatches the jobs to the corresponding slave nodes. Results are transferred to the application interface service. Application interface service is responsible to transfer the results to its upper layer. [4][7]

Algorithm for Master Mode

Step 1. The Application Interface Service segregates the application into smaller jobs.

Step 2. Jobs enter the queue. The Parameters are collected and a Parameter Table is constructed.

Step 3. Resource Finder is activated to find the availability of free resources in network, which can satisfy the conditional parameters along with the request packet.

Step 4. AHGL request Packet is broadcasted in the network for available resources in the network.

Step 5. The nodes on receiving the request furnish the information of free resources

Step 6. With the reply of free resources from nodes resource table is generated

Step 7. The Scheduler uses the Joint scheduling algorithm and congestion control to control the network traffic by calculating the urgency weight. The congestion control information is transferred to the transport layer.

Step 8. The Packet arrives the PDQ where urgency weight is calculated

Step 9. Based on the weight it is fed into dispatch table for further process

Step 10. The jobs allocated in the dispatch table are fed into the Dispatcher

Step 11. Dispatcher is Responsible for dispatching jobs to the corresponding slave nodes through ad hoc routing protocol.

Step 12. Dispatcher waits for the reply from slave nodes.

Step 13. The results obtained are sent to the Application Interface Service.

Step 14. Application interface service transfers the results to upper layer.

The AHGL slave mode is responsible for two services such as resource responder and job execution, which are shown in fig.5. The nodes in slave mode receive an AHGL request packet generated by the master. The slave nodes announce their free resources, which can fulfill the criteria and generates AHGL reply packet. They receive the packets with job description and have responsibility to execute the jobs dispatched by the AHGL master mode. After completion the job execution service transfers the job to the application layer, the result is transferred to the job execution service. Then the result is sent back to master mode.

Algorithm for Slave node

Step 1. Slave node receives the AHGL request packet generated by the master node.

Step 2. It generates a response packet when it satisfies the conditional parameters along with response packet filled with the characteristics of free resources sent to master node.

Step 3. Job execution service receives the jobs allocated by the master node.

Step 4. Job execution executes the allocated job on completion of job, the results are sent to job execution service.

Step 5. The results arrived are sent to master node.

6. Results and Discussions

The analysis on joint scheduling and congestion control in AHGL is done by implementation with NS-2 network simulator. The FindResource agent and ExecuteTask agent are derived from the Agent super class. The agent is responsible to track the specific characteristics of the node during its activation. The existing routing and other lower level protocols do this process.The simulation starts on each node with the FindResource agent.

The FindResource is responsible to flood the ad hoc network broadcasting a resource request packet, when the node is in master mode. The node receives the corresponding reply packet, the agent updates the resource table. The FindResource agent is activated in slave mode is programmed to construct a reply packet, which are satisfying the needed characteristics defined by the request packet. The reply packet is sent to the node in master mode and it continues the network flooding process. The tcl simulation class does the joint scheduling and congestion control. The tcl class identifies the channel rate, packet size and PDQ size. By using the channel rate for the packets in PDQ and depends upon the urgency it performs inter and intra scheduling to manage congestion

Table 1. AHGL Request Packet Fields

ID_{node}	SEQ	CPU	MEM	BAT
Node 1	52	2.5 GHz	512MB	70%
Node 2	61	3.0 GHz	250MB	60%

Table 2. AHGL Reply Packet Fields

ID_{node}	SEQ	CPU (GHz)	MEM	BAT	H	T_S (µs)	T_E (µs)	T_{EFF} (µs)
Node 1	52	3.0	512MB	75%	5	1.3	2.3	1
Node 2	61	3.5	512MB	75%	7	2.5	3.8	1.3

Table 3. AHGL Job Packet

ID_{app}	ID_{job}	ID_{sw}	L_{job}	N_{job}
7	25	VC++	3478	412
7	23	VC++	7532	359
7	21	VC++	4512	247

The nodes both in master and slave modes start the ExecuteTask agent. The agent gets the information of the next executable task from the schedulable table constructed by the joint scheduling and congestion control and resource table. It then segregates the task into several packets and sends them to their destination address in slave mode. The agent in master mode receives the replies from the slave nodes and checks the replies for task completion and allocate next task for the nodes in slave mode. When the ExecuteTask agent is in slave mode, it constructs the reply packets with result and sends to the master it calculates the delay to simulate the execution time of a task. First construct the topology for mobile ad hoc grid. Activate the FindResources agent in master mode to simulate the resource finder service. The tcl class for joint scheduling and congestion control is executed for simulating the scheduler activity. First construct the topology for mobile ad hoc grid. Activate the FindResources agent in master mode to simulate the resource finder service. The tcl class for joint scheduling and congestion control is executed for simulating the scheduler activity. At the end of the simulation activate the ExecuteTask agent to simulate the task execution service. The final results are processed robotically to acquire the parameters of the mobile ad hoc network. The flow rate set to this simulation process is 1 kbps and the maximum PDQ size is 60. The small packet size in queue is 15 bytes. The congestion control based on NUM problem is implemented in AHGL increments the network utility and control the traffic, which can speed up the job execution process in grid environment. Various transmission rates can be applied in the proposed system and the results are observed. The results for various transmission rates are shown in fig 6, fig. 7, fig 8 & fig 9. (Transmission rate T_r = Number of nodes/Number of packets).

Table 4. AHGL Result Packet

ID_{node}	ID_{app}	ID_{job}	Result
Node 1	7	25	51
Node 1	7	23	45
Node 1	7	21	78

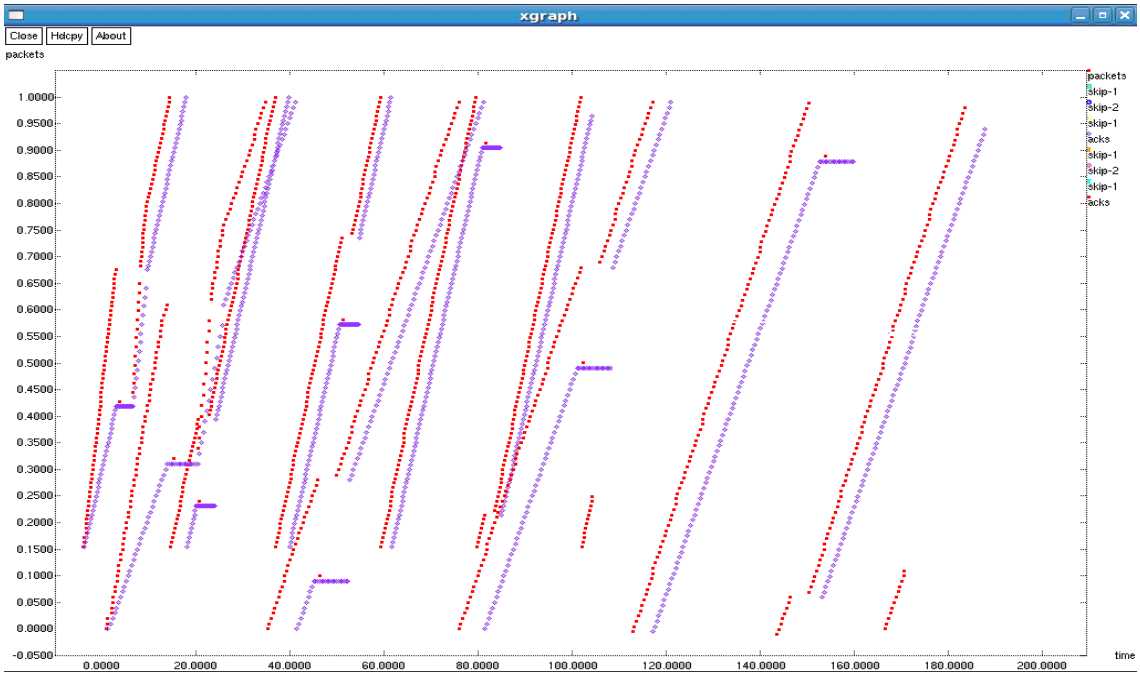

Figure 6. Packet Transmission with Congestion Control (T_r = 0.27)

First construct the topology for mobile ad hoc grid. Activate the FindResources agent in master mode to simulate the resource finder service. The tcl class for joint scheduling and congestion control is executed for simulating the scheduler activity.

At the end of the simulation activate the ExecuteTask agent to simulate the task execution service. The final results are processed robotically to acquire the parameters of the mobile ad hoc network.

The flow rate set to this simulation process is 1 kbps and the maximum PDQ size is 60. The small packet size in queue is 15 bytes. The congestion control based on NUM problem is implemented in AHGL increments the network utility and control the traffic, which can speed up the job execution process in grid environment. Various transmission rates can be applied in the proposed system and the results are observed. The results for various transmission rates are shown in fig 6, fig. 7, fig 8 & fig 9. (Transmission rate T_r = Number of nodes/Number of packets).

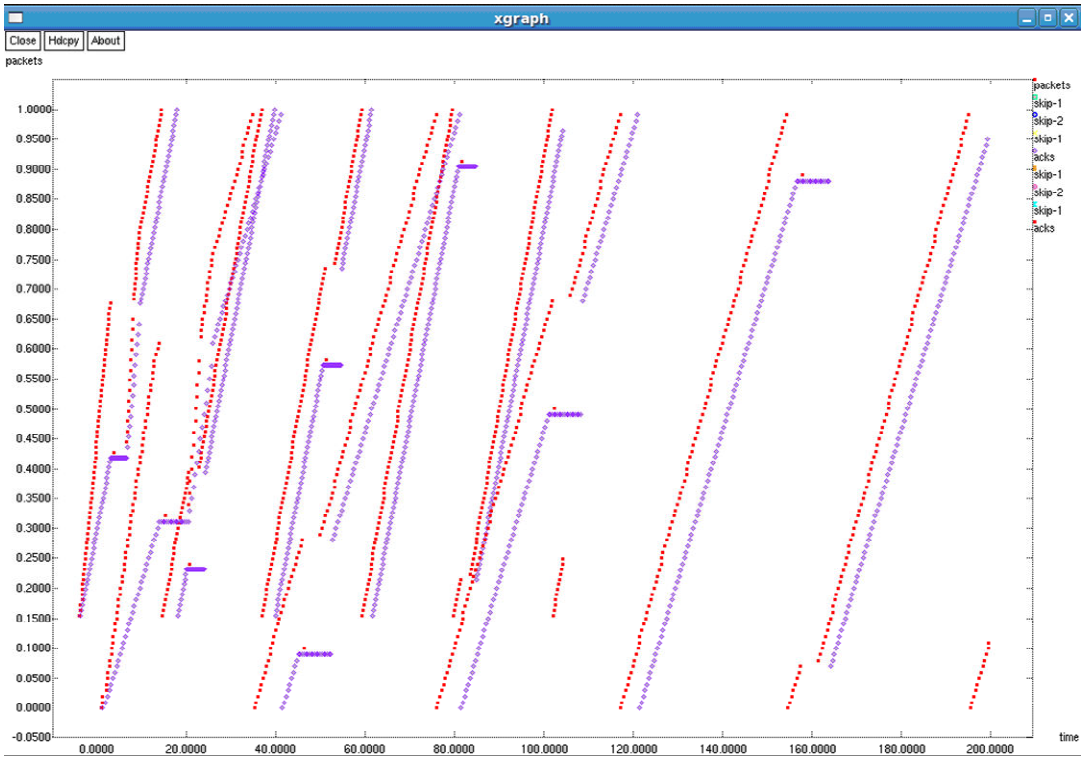

Figure 7. Packet Transmission with Congestion Control (T_r = 0.37)

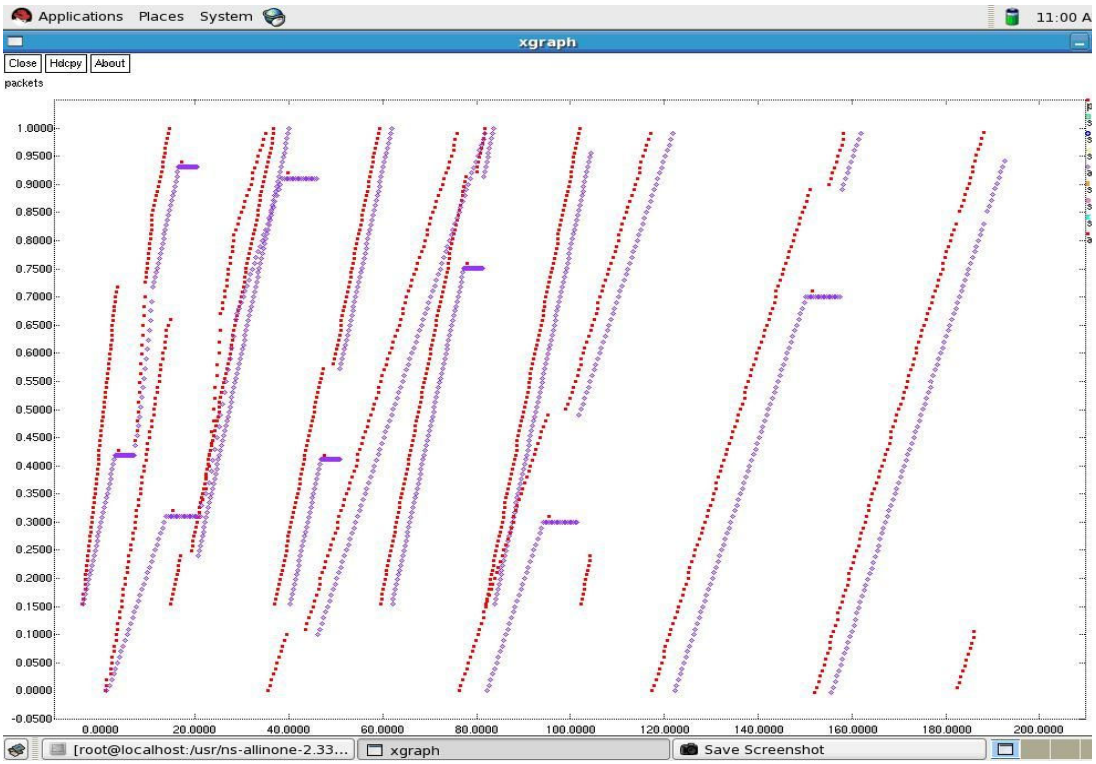

Figure 8. Packet Transmission with Congestion Control (T_r = 0.42)

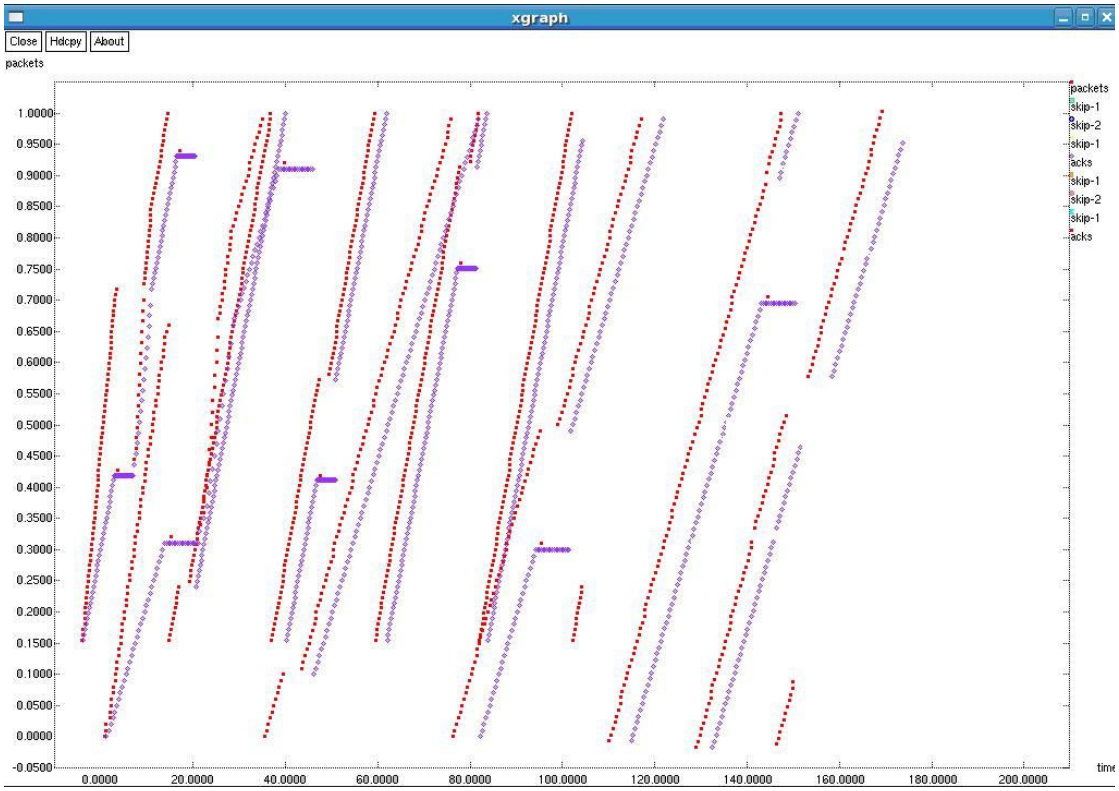

Figure 9. Packet Transmission with Congestion Control (T_r = 0.47)

Fig. 6 - 9 show that the number of packets transmitted through the mobile ad hoc grid environment is increased with respect to congestion control in network traffic and the observations are tabulated in Table 5. It also describes the hand over when the nodes are in mobility, the packet transmission time and time for hand over process are decreased, when the number of nodes in mobile ad hoc network is increased. The scheduling process in master mode with congestion control also reduces the over all execution time of jobs in mobile ad hoc network.

Table 5. Performance of AHGL with Joint Congestion control and Scheduling Algorithm

T_r	T_T (ms)	Packets Transmitted	No.of Handover	Average Time delay between handover (ms)
0.27	200	1000	10	200
0.37	180	1000	7	160
0.42	160	1000	7	140
0.47	140	1000	6	120

The number of nodes increased in mobile ad hoc grid environment decreases the execution time. When the number of nodes achieves the threshold level, the speedup is inundated. Figure 10 shows that the scalability and speed is enhanced in the implementation of congestion control in modified AHGL when compared with AHGL.

7. CONCLUSION

This paper presents a joint congestion control and scheduling algorithm implementation in AHGL of mobile ad hoc network. AHGL with congestion control is capable to adapt the dynamic environment and achieves high fault tolerance rate. This paper describes how the urgency weight influences the congestion in network traffic. This paper concludes the joint congestion . control and scheduling algorithm implemented in AHGL increases the scalability and speed of the job execution process in mobile ad hoc environment. This simulation work also achieves the maximum mobile ad hoc network utility.

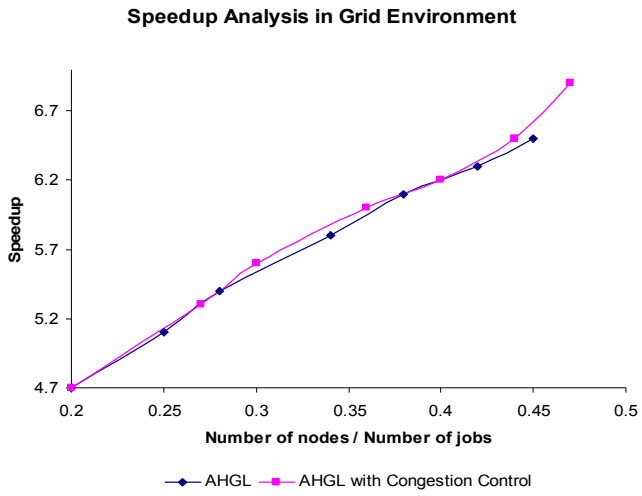

Figure 10. Speedup analysis between AHGL and Modified AHGL with congestion control

8. REFERENCES

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

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

[3] 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 on Intelligent Computing and Information Systems, pp:791-798,2009.

[4] Umut Akyol, Matthew Andrews, Piyush Gupta, John Honny, Iraj Saniee and Alexander Stolyar, "Joint Scheduling and Congestion Control in Mobile Ad Hoc Networks", Proc. IEEE INFOCOM , pp. 619–627,2008.

[5] Amira Soliman, Amr Kamel, Reem Bahgat, Walaa Sheta, "Semantic Grid Architecture for P2P Mobile Ad Hoc Networks", Fourth International Conference on Intelligent Computing and Information Systems, pp:365-373, 2009

[6] 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.

[7] Phillip M. Dickens, Vinod Kannan, "Application Level Congestion Control Mechanisms for Large Scale Data Transfers Across Computational Grids". In Proceedings of The International Conference on High Performance Distributed Computing and Applications, 2003.