Estimation of Resource usage in Peer to Peer Network

Amol P. Bhagat
Department of Computer Science and Engg, Prof. Ram Meghe College of Engineering and Management, Badnera, India

Pravin Malve
Department of Computer Science and Engg, Governmentant Polytechnic Murtizapur, Amravati

Jayant Mehare
Department of Information Technology, Sipna College of Engineering, Amravati

ABSTRACT
Grid systems allow us to take advantage of available resources lying over a network. However, these systems impose several difficulties to their usage (e.g. heavy authentication and configuration management); in order to overcome them, Peer-to-Peer systems provide open access making the resources available to any user. A device in a P2P network can provide access to any type of resource that it has at its disposal, whether documents, storage capacity, computing power, or even its own human operator. This paper demonstrates the concept of resource scheduling and request forwarding in peer to peer network.

General Terms
Networking, Scheduling.

Keywords
Network Protocols, Resource Scheduling, Peer to Peer Network, Network Simulation, Network Animator.

1. INTRODUCTION
Peer-to-Peer [1] systems are characterized by their ability to function, scale, and self-organize in the presence of highly transient population of failure-prone nodes. The great advantage of this approach over other models is the no dependence on centralized servers, which suffer from problems such as bottlenecks, single points of failure, among other.

Peer-to-Peer (P2P) technology enables any network-connected device to provide services to another network-connected device. A device in a P2P network can provide access to any type of resource that it has at its disposal, whether documents, storage capacity, computing power, or even its own human operator. The device in a P2P network could be anything ranging from a super computer to simple PDA. P2P technology is a robust and impressive extension of the Internet's philosophy of robustness through decentralization.

The main advantage of P2P networks is that it distributes the responsibility of providing services among all peers on the network; this eliminates service outages due to a single point of failure and provides a more scalable solution for offering services.

1.1 Resource Management Model
Client/server solutions rely on the addition of costly bandwidth, equipment, and co-location facilities to maintain a robust solution. P2P can offer a similar level of robustness by spreading network and resource demands across the P2P network. Several different P2P architectures have been proposed so far, a comprehensive survey is provided in [1]. The job distribution and management in network is carried out as shown in Figure 1 where a machine, acting as a resource consumer, distributes tasks among available machines, resource providers, in order to perform a CPU-intensive job demanded by a user.

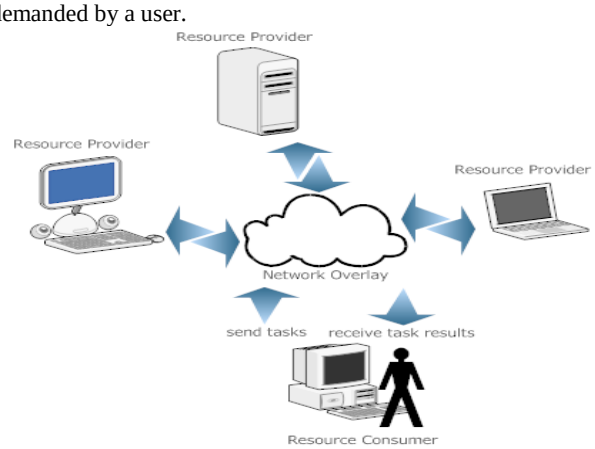

Fig 1. Resource Management Model

Resource providers receive the tasks, compute them, and send the results back to the consumer node (the job holder). All machines are connected through an overlay network, which is built on top of another network (i.e. Internet) and provides services of routing and lookup.

.

2. LITERATURE REVIEW
Peer-to-Peer has been gaining a huge success across the Internet. Such architectures are designed for the direct sharing of computer resources (CPU cycles, storage, and content) rather than requiring the intermediation of a centralized server or authority [2].

Deeds [3, 11] is a history-based access control system whose policies must be written in Java. It is useful to provide security in P2P network. For resource discovery Iamnitchi et al [4] have compared different searching methods. Cheema et al [5] proposed a solution for exploiting the single keyword lookup for CPU cycle sharing systems. Globus [6] is an enabling technology for grid deployment. It provides mechanisms for communication, authentication, network information, data access, amongst other.

Condor [7] allows the integration and use of remote workstations. It maximizes the utilization of workstations and expands the resources available to users, by functioning well in an environment of distributed ownership. BOINC [8] is a platform for volunteer distributed cycle sharing based on the client-server model. It relies on an asymmetric relationship where users, acting as clients, may donate their idle CPU cycles to a server, but cannot use spare cycles, from other clients, for themselves.

CCOF [9] is an open peer-to-peer system seeking to harvest idle CPU cycles from its connected users. OurGrid [10] is a peer-to-peer network of sites which tries to facilitate the inter-domain access to resources in a equitably manner.

Although the applications of P2P technologies are fascinating and still much more is left to be discovered but there are some challenges that are to be faced. Unfortunately, the current applications of P2P tend to use protocols that are incompatible in nature, reducing the advantage offered by gathering devices into P2P networks. Each network forms a closed community, completely isolated from the other networks and incapable of using their services. So, the requirement of a common platform which can bind all the peers and facilitates free communication between them is pre-requisite for P2P to realize its full potential.

2.1 Implementation

There are various tools available for simulating different network models. Ns2/ns3, OPNET and NetSim are some of the tools that can be used for the simulation of the various network architectures and models. The ns-2 simulator is a discrete-event network simulator targeted primarily for research and educational use. The ns-2 is written in C++. ns-2 is open-source, and the project strives to maintain an open environment for researchers to contribute and share their software.

Ns-2 is scripted in OTcl and results of simulations can be visualized using the Network Animator nam. It is not possible to run a simulation in ns-2 purely from C++ (i.e., as a main() program without any OTcl). Moreover, some components of ns-2 are written in C++ and others in OTcl. Considering these features of ns-2 **ns-allinone-2.34** is used for the implementation of the proposed dissertation work. Initially **ns-allinone-2.34** is downloaded from [15]. NS-2 is designed to run from on most UNIX based operating systems. It is possible to run NS-2 on Windows machines using Cygwin. If you don't have a UNIX install, you can also use a virtual linux machine and run that under Windows. In the dissertation work the **Fedora core 13** operating system is used for installation and configuration of the **ns-2.34**. The **ns-2.34** is configured on the path /home/project/Desktop/project/. For configuring the ns the following commands are executed in the terminal. Before configuration we should make sure that we have standard development packages like 'make' and 'gcc'.

```
tar -xzf ns-allinone-2.34.tar.gz
cd ns-allinone-2.34
./install
```

After the execution of the above commands on terminal if everything is fine without any errors then we will get following messages on the terminal.

```
Ns-allinone package has been installed successfully.
Here are the installation places:
tcl8.4.11:/home/amol/Desktop/project/ns-allinone-2.34/{bin,include ,lib}
tk8.4.11:/home/amol/Desktop/project/ns-allinone-2.34/{bin, include, lib}
otcl:       /home/amol/Desktop/project/ns-allinone-2.34/otcl-1.11
tclcl:      /home/amol/Desktop/project/ns-allinone-2.34/tclcl-1.17
ns:         /home/amol/Desktop/project/ns-allinone-2.34/ns-2.29/ns
nam:        /home/amol/Desktop/project/ns-allinone-2.34/nam-1.11/nam
xgraph:  /home/amol/Desktop/project  /ns-allinone-2.34/xgraph-12.1
gt-itm:     /home/amol/Desktop/project/ns-allinone-2.34/itm,    edriver,    sgb2alt, sgb2ns, sgb2comns, sgb2hierns

----------------------------------------
----------------------------------------

Please put /home/amol/Desktop/project/ns-allinone-
2.34/bin:/home/amol/Desktop/project/ns-allinone-2.34/tcl8.4.18       /unix:/home/amol/Desktop/project/ns-allinone-
2.34/tk8.4.18/unix


into   your   PATH   environment;   so   that   you'll      be      able      to      run itm/tclsh/wish/xgraph.

IMPORTANT NOTICES:

(1)          You          MUST          put /home/amol/Desktop/project/ns-allinone-
2.34/otcl-
1.13:/home/amol/Desktop/project/ns-allinone-2.34/lib,

    into your LD_LIBRARY_PATH environment variable.
    If it complains about X libraries, add path to your X libraries
    into LD_LIBRARY_PATH.
    If you are using csh, you can set it like:
                setenv     LD_LIBRARY_PATH <paths>
    If you are using sh, you can set it like:
                export LD_LIBRARY_PATH=<paths>
(2)          You          MUST          put /home/amol/Desktop/project/ns-allinone-
2.34/tcl8.4.18/library      into      your TCL_LIBRARY environmental
    variable.   Otherwise   ns/nam   will complain during startup.
(3) [OPTIONAL] To save disk space, you can now delete directories tcl8.4.11
    and tk8.4.11. They are now installed under       /home/amol/Desktop/project/ns-allinone-2.34/{bin,include,lib}
After these steps, you can now run the ns validation suite with
cd ns-2.34; ./validate
For trouble shooting, please first read ns problems page
http://www.isi.edu/nsnam/ns/ns-problems.html. Also search the ns mailing list archive
for related posts.
```

For the validation of the installation we can run the next command on terminal as

```
cd ns-2.34
./validate
```

After successful installation we need to modify some of the environment variables namely PATH, LD_LIBRARY_PATH and TCL_LIBRARY. To accomplish this the following commands are executed on the terminal.

```
export set PATH=$PATH:/home/amol/Desktop/project/ns-allinone-2.34/bin:/home/amol/Desktop/project/ns-allinone-2.34/tcl8.4.18/unix:/home/amol/Desktop/project/ns-allinone-2.34/tk8.4.18/unix
export set LD_LIBRARY_PATH=/home/amol/Desktop/project/ns-allinone-2.34/otcl-1.13:/home/amol/Desktop/project/ns-allinone-2.34/lib
export set TCL_LIBRARY=/home/amol/Desktop/project/ns-allinone-2.34/tcl8.4.18/library
```

After the successful configuration of ns environment for the implementation of the proposed dissertation work initially we have created **rc (resource consumer)** package inside **ns-2.34** which defines various properties of the data in terms of packet which are transferred to and from various node which can be identified as packets which are using implemented **resource consumer** protocol for intrusion detection in the network. The implemented algorithm which is written inside **rc.cc** is situated inside the router through which packets are transferred and filtered. It also includes different parameters which are defined for the implementation of the resource scheduling system which includes different types of packet which is transferred between different nodes. There may be number of resources which are present inside the network providing different services. Scheduling of various requests on the particular resource service provider is handled through this code. Inside rc we have defined four files **rc.cc, rcPacket.cc, rc.h and rcPacket.h**. After adding rc into **ns-2.34** we need to modify some of the files of ns environment which are **ns-2.34/common/packet.h**, **ns-2.34/tcl/lib/ns-packet.tcl** and **ns-2.34/Makefile.** After these changes we need to again execute make command on the terminal to reflect the changes in the ns environment. We have demonstrated the result of implemented work through various simulations implemented in terms of tcl script **simgrid.tcl** demonstrates resource scheduling in the network with single resource consumer and three resource provider. In **simgridfinal.tcl** we have demonstrated the resource scheduling and utilization in the network with three resource consumer and three resource providers. The network with multiple resource consumers and resource providers is demonstrated in **simgridmax.tcl**.

3. EXPERIMENTATION AND RESULTS

The results are carried out by different simulations which are implemented to demonstrate the different ways of resource scheduling and utilization in peer to peer network in different types of network architecture. Initially simgrid.tcl is implemented to demonstrate resource utilization in network where there are three types of resource nodes and one resource consumer node. For the execution of this tcl script initially all the environment variables are set and the following command is executed on the terminal.

```
ns simnidsnewmorenodes.tcl
Sending request:1
Node 2: Request received... Now forwarding to grid node 3 for execution
Node 2: Request received... Now forwarding to grid node 3 for execution
Node 3: Serving request of type 1
It took 0.223400 seconds to service request.
Average time taken 0.127412.
Sending request:1
Node 2: Request received... Now forwarding to grid node 3 for execution
Node 2: Request received... Now forwarding to grid node 3 for execution
Node 3: Serving request of type 1
It took 0.346800 seconds to service request.
Average time taken 0.127412.
Sending request:5
Node 2: Grid serving request of type 5
It took 0.470200 seconds to service request.
Average time taken 0.123400.
Sending request:2
Sending request:3
Node 2: Request received... Now forwarding to grid node 4 for execution
Node 2: Grid serving request of type 3
It took 0.593600 seconds to service request.
Average time taken 0.123400.
Node 4: Serving request of type 2
It took 0.717000 seconds to service request.
Average time taken 0.124610.
Sending request:2
Node 2: Request received... Now forwarding to grid node 4 for execution
Node 4: Serving request of type 2
It took 0.840400 seconds to service request.
Average time taken 0.124610.
Sending request:3
Node 2: Grid serving request of type 3
It took 0.963800 seconds to service request.
Average time taken 0.123400.
Sending request:2
Node 2: Request received... Now forwarding to grid node 4 for execution
Node 4: Serving request of type 2
It took 1.087200 seconds to service request.
Average time taken 0.124610.
Sending request:4
Node 2: Grid serving request of type 4
It took 1.210600 seconds to service request.
Average time taken 0.123400.
Sending request:4
Sending request:3
Node 2: Grid serving request of type 4
It took 1.334000 seconds to service request.
Average time taken 0.123400.
Node 2: Grid serving request of type 3
It took 1.457400 seconds to service request.
Average time taken 0.123400.
```

```
Sending request:1
Node 2: Request received... Now
forwarding to grid node 3 for execution
Node 2: Request received... Now
forwarding to grid node 3 for execution
Node 3: Serving request of type 1
It took 1.580800 seconds to service
request.
Average time taken 0.127412.
Sending request:2
Node 2: Request received... Now
forwarding to grid node 4 for execution
Node 4: Serving request of type 2
It took 1.704200 seconds to service
request.
Average time taken 0.124610.
Sending request:3
Node 2: Grid serving request of type 3
It took 1.827600 seconds to service
request.
Average time taken 0.123400.
Sending request:3
Node 2: Grid serving request of type 3
It took 1.951000 seconds to service
request.
Average time taken 0.123400.
```

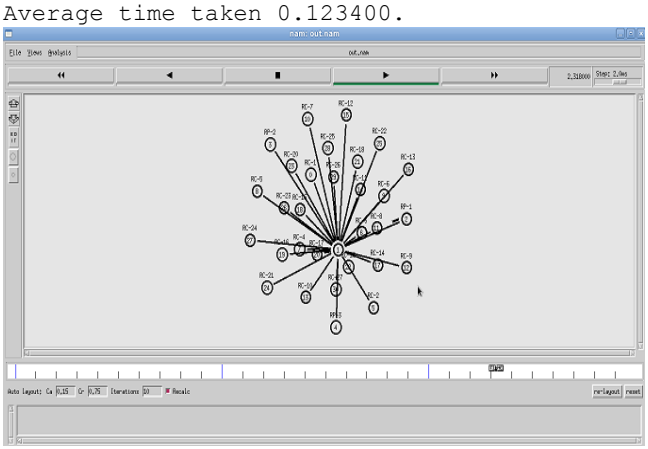

Fig 2: Request forwarded to by resource provider

The following table shows the comparison of the various requests that are received and serviced by the different nodes.

Table 1. Request received and serviced by the available nodes

Nodes	Number of Requests Received	Number of Requests Serviced	Total time on the node (Seconds)	Actual Resource Utilization (Seconds)
Node 2	14	6	1.580800	0.49844
Node 3	2	2	1.087200	0.254824
Node 4	4	4	1.580800	0.7404

As it can be observed from the table the node 4 is utilized more than the remaining two nodes. It also shows the total time for which the resource was in the active state in the network and its actual utilization in the network. So here the total resource usage is estimated in the network.

4. CONCLUSIONS

The resource scheduling in the peer to peer network is demonstrated in this paper. The algorithm is implemented for request forwarding when a particular type of resource is not available on the requested node. The nodes actually process less number of requests as compared to the number of request received. The work can be further extended to dynamically scheduling the request on the various types of resources. The implemented work also exploits the parallel execution of multiple requests in the network.

5. REFERENCES

[1] Pourebrahimi B., Bertels K., Vassiliadis S. A Survey of Peer-to-Peer Networks. *Technical Report, Computer Engineering Laboratory, ITS, TU Delft, The Netherlands. 2004*.

[2] S. Androutsellis-Theotokis and D. Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys (CSUR), 36(4):335–371, December 2004*.

[3] G. Edjlali, A. Acharya, and V. Chaudhary. History-based access control for mobile code. *In CCS '98: Proceedings of the 5th ACM conference on Computer and communications security, pages 38–48, New York, NY, USA, 1998. ACM*.

[4] A. Iamnitchi and I. Foster. A peer-to-peer approach to resource location in grid environments. *In Grid resource management: state of the art and future trends, pages 413–429, Norwell, MA, USA, 2004. Kluwer Academic Publishers*.

[5] A. S. Cheema, M. Muhammad, and I. Gupta. Peer-to-peer discovery of computational resources for grid applications. *In GRID '05: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing, pages 179–185, Washington, DC, USA, 2005. IEEE Computer Society*.

[6] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications, 11:115–128, 1997*.

[7] M. Litzkow, M. Livny, and M. Mutka. Condor - a hunter of idle workstations. *In Proceedings of the 8th International Conference of Distributed Computing Systems, June 1988*.

[8] D. P. Anderson. Boinc: A system for public-resource computing and storage. *In GRID '04: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, pages 4–10, Washington, DC, USA, 2004. IEEE Computer Society*.

[9] V. Lo, D. Zhou, Y. Liu, and S. Zhao. Cluster computing on the fly: P2p scheduling of idle cycles in the internet. *In the internet, 3rd International Workshop on Peer-to-Peer Systems (IPTPS 2004), pages 227–236, 2004*.

[10] N. Andrade, W. Cirne, F. Brasileiro, and P. Roisenberg. Ourgrid: An approach to easily assemble grids with equitable resource sharing. *In Proceedings of the 9th Workshop on Job Scheduling Strategies for Parallel Processing, Seattle, WA, USA, June 2003*.

[11] S´ergio Esteves, Lu´ıs Veiga and Paulo Ferreira GridP2P: Resource Usage in Grids and peer-to-Peer Systems. *INESC-ID/IST, Distributed Systems Group, Rua Alves Redol, 9, 1000-029 Lisboa, Portugal 2010 IEEE.*