ISSN No. 0976-5697

Volume 2, No. 4, July-August 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Request Forwarding in Peer to Peer Network

Pravin U. Malve
Student, (M.E.) SCOET,
SGBAU University,
Amravati, Maharashtra, India
pravin.malve29@gmail.com

Prof. V.S. Gulhane*
Associate Professor, SCOET,
SGBAU University,
Amravati, Maharashtra, India
v_gulhane@rediffmail.com

Abstract: Peer to Peer Systems distributes the responsibility of providing services among all peers on the network; this eliminates service outages due to a single point of failure, 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. Resource providers receive the tasks, compute them, and send the results back to the consumer node (the job holder). This paper demonstrates the concept of request forwarding in peer to peer network.

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

I. INTRODUCTION

Peer-to-Peer 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.

In addition, P2P networks exploit available bandwidth across the entire network by using a variety of communication channels and by filling bandwidth up to the brim of the Internet. Unlike traditional client/server communications, in which specific routes to popular destinations can become overloaded (for example, the route to google.com), P2P enables communication via a variety of network routes, thereby reducing network overloading. P2P has the capability of serving resources with high availability at a much lower cost while maximizing the use of resources from every peer connected to the P2P network. 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. 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

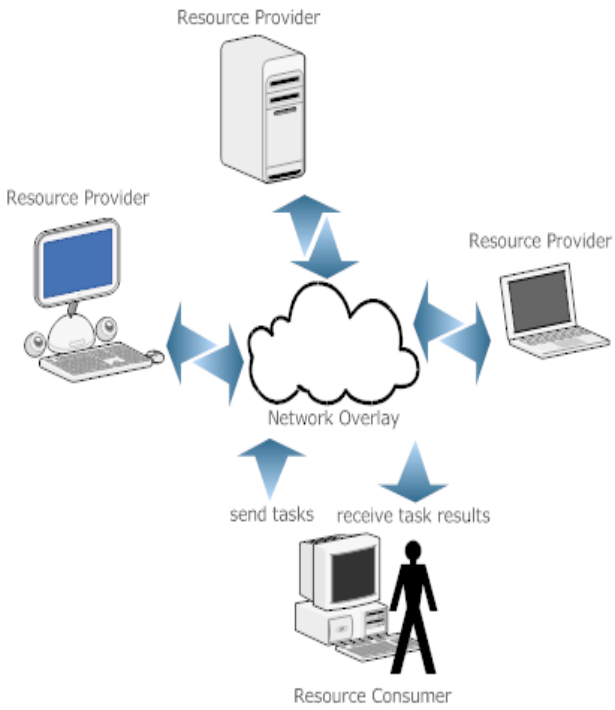


Figure. 1: Resource Management Model

II. 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].

Currently, not only scientists, but also typical non-expert computer users are willing to perform intensive tasks on their computers. However, these tasks could be quite different, like: compressing a movie file, generating a complex image from a specification, compacting large files, among other. More precisely, these tasks consume a relatively large amount of time and memory, delaying other processes that are running at the same time. Along the way, one becomes bored and impatient. From another point of view, there are many Internet connected computers around the world whose resources are not fully utilized. Most of the time, non-expert users have just some low CPU-intensive processes running on their machines, therefore giving a sense of waste.

Given the current context, we intend to deploy a platform where any ordinary user may consume and provide resources, namely idle CPU cycles, over a dynamic network that could be local or wide (e.g. Internet), in order to speed up common, and widely used, applications which are CPU-intensive. There are two fundamental requirements: first is while we intend to exploit parallel execution in desktop applications, the system must ensure a fine-grained control over the shared resources, and second applications should be kept unmodified in order to take advantage of all the software already existing.

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 [2] proposed a solution for exploiting the single keyword lookup for CPU cycle sharing systems. Globus [5] 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 [3] 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 [12] is an open peer-to-peer system seeking to harvest idle CPU cycles from its connected users. OurGrid [8] is a peer-to-peer network of sites which tries to facilitate the inter-domain access to resources in a equitably manner.

III. 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. 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.

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 request forwarding in the network with single resource consumer and three resource provider.

IV. EXPERIMENTATION AND RESULTS

Simgrid.tcl is implemented to demonstrate request forwarding 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

A. ns simgrid.tcl

After the execution **out.nam** file is created inside the current working directory and we get the following output on the terminal.

Sending request: 1

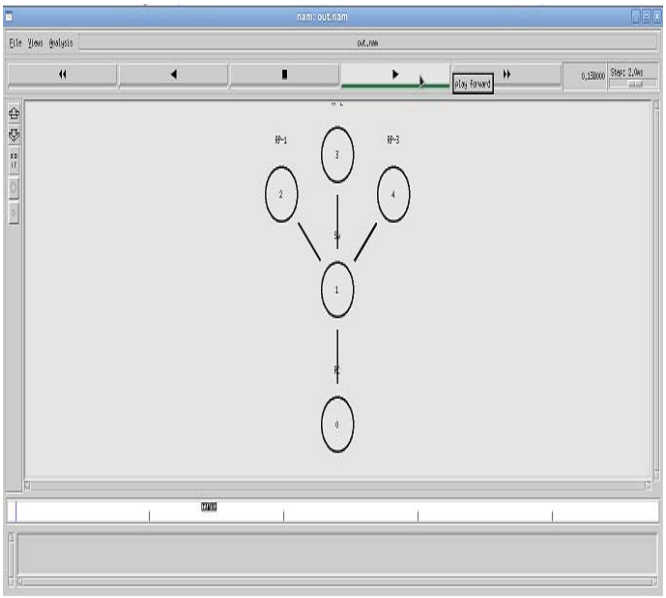
Node 2: Request received... Now forwarding to grid node 3 for execution
Node 2: Request received... Now forwarding to grid node 3 for execution
Node 3: Serving request of type 1
It took 0.223400 seconds to service request.
Average time taken 0.127412.
Sending request:2
Node 2: Request received... Now forwarding to grid node 4 for execution
Node 4: Serving request of type 2
It took 0.346800 seconds to service request.
Average time taken 0.124610.
Sending request: 3
Node 2: Grid serving request of type 3
It took 0.470200 seconds to service request.
Average time taken 0.123400.

It can be observed from the output various request has been generated from resource consumer. These requests are for different types of resources in the network. The initial request is forwarded to node 2 but this request is for type 1 resource, this type of resource is not available at node 2, hence the request is forwarded to node 3 as resource of type 1 is available at this node the request get serviced. The result also shows the actual time required servicing the request and the total time required to process the request from its initialization. If any of the resource is not available then request is continuously transferred in the network till that type of resource doesn't available.

We can run the simulation by executing the following command on the terminal.

B. nam out.nam

After execution the output is generated inside the network animator. The results are shown below.



Figureg 2: Initial state of network with single resource consumer and three resource provider.

As shown in the following figure 2 the request is initialized from the resource consumer and it is forwarded to node 2 as shown in figure 3. But the requested resource is not available at this node thus the request is forwarded to node 3 as type 1 resource is available at this node as shown in figure 4.

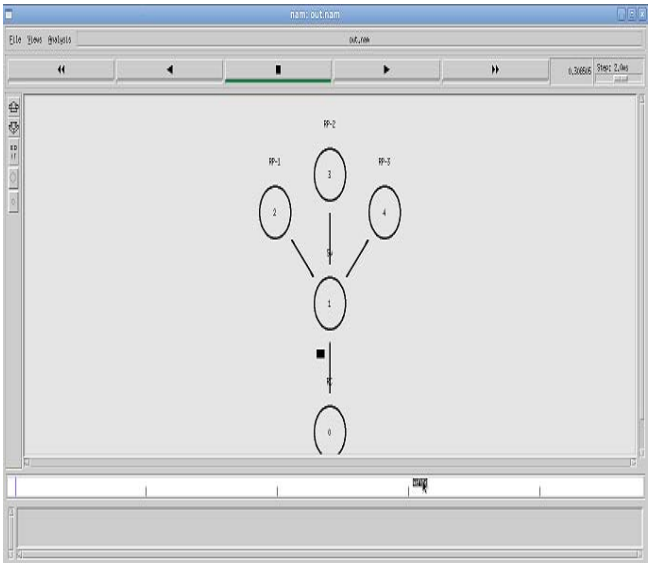


Figure 3: Request initialize by the resource consumer node.

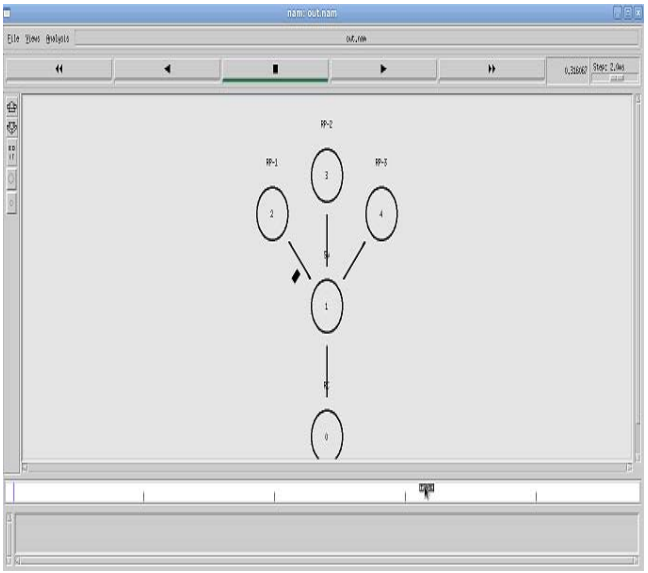


Figure 4: Request sent to node 2 by the resource consumer node

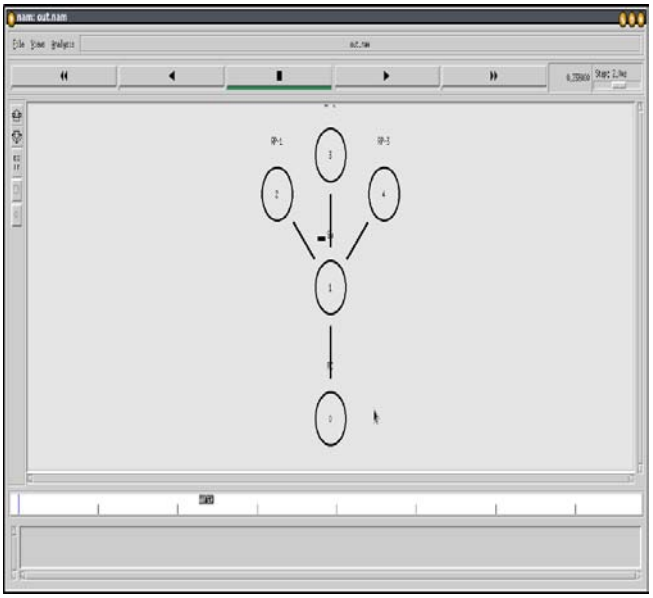


Figure 5: Request forwarded to node 3 as resource is not available at node 2.

V. CONCLUSION

This paper gives various aspects of implementing the system for request forwarding in peer to peer network. 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. but as the number requests send at a heavy node increases the number of request processed increases.

VI. REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

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