International Journal of Networking
ISSN: 2249-278X & E-ISSN: 2249-2798, Volume 2, Issue 1, 2012, pp.-25-27.
Available online at http://www.bioinfo.in/contents.php?id=108

RESOURCE SCHEDULING IN THE PEER TO PEER NETWORK

MALVE P.U.[1]* AND GULHANE V.S.[2]

[1]Department of Computer Engineering, Govt. Polytechnic, Murtizapur-444107, MS, India.
[2]Department of CSE, Sipna' COET, Amravati-444606, MS, India.
*Corresponding Author: Email- pravin.malve29@gmail.com

Received: February 21, 2012; Accepted: March 15, 2012

Abstract- An efficient resource discovery mechanism is one of the fundamental requirements for Peer to peer computing systems, as it aids in resource management and scheduling of applications. Resource discovery activity involves searching for the appropriate resource types that match the user's application requirements. Various kinds of solutions to resource discovery have been suggested, including centralized and hierarchical information server approaches However, both of these approaches have serious limitations in regard to scalability, fault tolerance, and network congestion. This paper describes purposed Resource scheduling algorithm that can be used for efficient resource discovery in peer to peer network.

Keywords- Peer to Peer Network, Resource Scheduling, Network Animator, ns-2 Simulation, Resource discovery

Citation: Malve P.U. and Gulhane V.S. (2012) Resource Scheduling in The Peer to Peer Network. International Journal of Networking, ISSN: 2249-278X & E-ISSN: 2249-2798, Volume 2, Issue 1, pp.-25-27.

Introduction

An efficient resource discovery mechanism is one of the fundamental requirements for peer to peer computing systems, as it aids in resource management and scheduling of applications. 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 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..

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].

Related work

For Resource discovery, Lamnitchi et al [1] have compared different searching methods. it turned out that a learning -based strategy achieves more performance. It consists of forwarding a request to the node that answered similar requests previously (i.e. Using possibly large cache.

Moreover results have shown that searching mechanisms which keep a history of past events than the ones that do not store any information about node. Cheema et al [2] purposed a solution for exploiting the single keyword DHT lookup for CPU cycle sharing systems. This solution consists in encoding resource identifiers based on static and dynamic resource descriptions. The static ones could be, for instance, the OS configuration, RAM, or CPU speed. While dynamic descriptions are related to the availability levels of resources, such as a percentage of idle CPU. With this encoding mechanism, it is possible to create mapping between resource and node identifiers in structured peer-to-Peer networks. Paredes [3] presents a solution through which this queries are forwarded to the neighbor nodes with the best availability and reputation.

BOINC[4] is a platform for volunteer distributed cycle sharing based on the client-server model. CCOF [5] is an open peer-to-peer system seeking to harvest idle CPU cycles from its connected users. OurGrid [6] is a peer-to-peer network of sites which tries to facilitate the inter-domain access to resources in a equitably manner.

Resource scheduling algorithm

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.

```
#include "rc.h"
#include "rcPacket.h"
#include "cmu-trace.h"
#include "ip.h"
#include <cstdlib>
#include <stdlib.h>
#include <math.h>
//This is important: It provides interface for tcl script
static class rcClass: public TclClass
{
public:
 rcClass ( ): TclClass("Agent/Rc") {}
        TclObject* create ( int, const char*const*) {
                return (new Rc ( ));
        }} class_rc;
Rc::Rc ( ) : Agent(PT_RC), RcTimer (*this)
{}
 void Rc::recv (Packet* p, Handler*)
{ //create ip packet pointer and read IP header received packet==>
necessary to find position of the node (left or right)
        hdr_ip *ip = hdr_ip::access(p);
//create rc pointer and read ipke header of received packet
        hdr_rc* my= hdr_rc::access(p);
 // Discard the packet //necessary as now packet is processed
//printf ("Packet Received\n");
        //Create Packet
        Packet* q = allocpkt ( );
        //create IP header for our packet
        hdr_ip *ip_q = hdr_ip::access(q);
//create communication header for our packet:
        hdr_cmn* cmn_q = hdr_cmn::access (q);
        //now create our IPKE packet
        hdr_rc* my_q= hdr_rc::access(q);
        switch (ip->daddr()) {
case 2:
//check resource type
        switch (my->resourceType)
{ case 1:
        //send Request to node 3
 my_q->resourceType = my->resourceType;
        cmn_q->size ( ) = 100;
        cmn_q->ptype ( ) = PT_RC;
        ip_q->daddr ( )= 3;
        ip_q->dport ( )= ip->dport ( );
printf ("\t Node 2: Request received... Now forwarding to grid node
3 for execution\n");
 send (q, 0);
        //Packet::free (q);         break;
case 2:
 my_q->resourceType = my->resourceType;
//set size: may be wrong not needed in our case
        cmn_q->size ( ) = 100;
        cmn_q->ptype ( ) = PT_RC;
        ip_q->daddr ( )= 4;
//now generate unique port and use it; ns2 standard              ip_q
->dport ( )= ip->dport ( );
printf ("\t Node 2: Request received... Now forwarding to grid node
4 for execution\n");
//send request to node 4
send (q,0); break;
default: //server request
printf ("\t Node 2: Grid serving request of type %d\n", my-
>resourceType);
break;
} break;
case 3:
//serve request
printf("\tNode 3: Serving request of type 1\n"); break;
        case 4: //serve request
printf("\tNode 4: Serving request of type 2\n");
        break; default:
printf ("\tNode %d:Packet received\n", ip->daddr());
                        break; }
//      Packet::free (q);
        Packet::free (p); }
int Rc::command(int argc, const char*const* argv)
{ //This is executed when command "Request" is received from
node
if (strcmp (argv[1], "Request") == 0) {
                //Create Packet
                Packet* p = allocpkt ( );                 //create   IP
header for our packet
        hdr_ip *ip = hdr_ip::access (p);
        hdr_cmn*cmn = hdr_cmn::access (p);
        //now create our IPKE packet
        hdr_rc* my= hdr_rc::access (p);                          /   /
Our request data
        my->resourceType = atoi (argv[2]);
        //set size: may be wrong not needed in our case
 cmn->size ( ) = 200;
 cmn->ptype() = PT_RC;
        ip->daddr()= 2;
        ip->dport()= here_port_;
printf ("Sending request:%d\n", my->resourceType);
                        send (p, 0);
return (TCL_OK); }
return (Agent::command(argc, argv)); }
void rcTimeout( ) {
//not needed
```

Implementation

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,

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). Considering these features of ns-2 ns-allinone-2.34 is used for the implementation of the proposed Scheduling algorithm.

NS-2 is designed to run from on most UNIX based operating systems. In the purposed work the Fedora core 13 operating system is used for installation and configuration of the ns-2.34. The purposed Scheduling algorithm demonstrated scheduling of various tasks having a peer-to-peer network containing One resource consumer and Three resource providers.

For the execution of this tcl script the following command is executed on the terminal.

```
ns simgrid.tcl
```

After the execution out.nam file is created 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 3: Serving request of type 1
It took 0.223400 seconds to service request.
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.
Sending request:3
Node 2: Grid serving request of type 3
It took 0.470200 seconds to service request.
```

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

```
nam out.nam
```

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

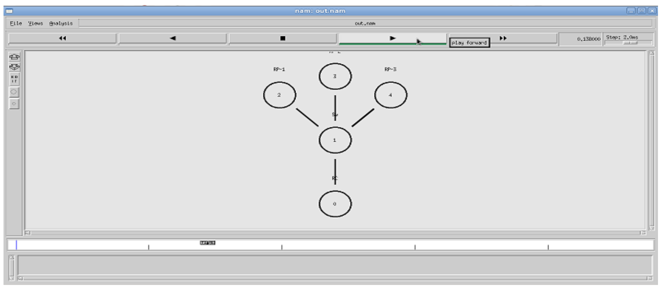

Fig. 1- One resource consumer and Three resource providers.

Conclusion

It can be observed from the output various requests 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 If any of the resource is not available then request is continuously transferred in the network till that type of resource doesn't available.

References

[1] Iamnitchi A. and Foster I. *Grid resource management: state of the art and future trends*, 413- 429

[2] Cheema A.S., Muhammad M. and Gupta I. (2005) *The 6th IEEE/ACM International Workshop on Grid Computing*, 1790-185.

[3] Paredes F.R. (2008) *Topologies de overlays peer to peer para descoberta de recursos*., *Master's thesis, Institute Technio*.

[4] Anderson D.P. *The 5th IEEE/ACM International Workshop on Grid Computing*, 4-10.

[5] Lo V., Zhou D., Liu Y. and Zhao S. (2004) 3*rd International Workshop on Peer-to-Peer Systems*, 227-236.

[6] Andrade N., Cirne W., Brasileiro F. and Roisenberg P. (2003) *The 9th Workshop on Job Scheduling Strategies for Parallel Processing*.