Comparative analysis of resource utilization in peer to peer networks

Pravin U. Malve¹, V.S. Gulhane², A.V. Deorankar³ and R.S. Mawale⁴
¹Govt. Polytechnic, Murtizapur, Akola, Maharashtra.
²Sipna, COET, Amravati, Maharashtra.
³Govt. College of Engineering, Amravati, Maharashtra.

ARTICLE INFO
Article history:
Received: 28 January 2013;
Received in revised form: 21 March 2016;
Accepted: 26 March 2016;

Keywords
Network animator,
Network simulator,
Peer to peer networks (P2P),
Resource consumer,
Resource provider.

ABSTRACT
P2P networks has been used over the years to overcome the problem of node failure & service availability in the client server network but it is found that the some nodes in P2P networks actually process less number of requests as compared to the number of request forwarded to a particular node. But as the number of requests send at a node increases the number of request processed at a particular node increases. This paper presents the comparative analysis of resource utilization in peer to peer networks.

© 2016 Elixir All rights reserved.

Introduction
In a peer to peer network a machine, acts as a resource consumer or resource provider or both, depending on the needs of a user. Resource providers receive the tasks, compute them, and send the results back to the consumer node. A node wants to provide its services to a network acts as resource provider.

All the nodes serve their resources for servicing requests but still some of the resources remains underutilized.

Given the current context, we intend to deploy a platform where any ordinary user increases the number of requests sent at a node so that the number of request processed at a particular node increases.

Literature Review
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 nodes, such as a random walk.

Rozлина Mohamed [3] presents the Taxonomy for Peer-to-Peer resource discovery. 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 [4] presents a solution through which these queries are forwarded to the neighbor nodes with the best availability and reputation. The best availability concerns the idleness level of a node in terms of its resources, and the reputation consists in the capability of a node to forward a query to other nodes with availability. The results have shown that this approach is efficient and scalable and thus mapping to peer-to-Peer requirements.

Cycle sharing systems:
Condor [5] 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 [6] 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 [7] 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.

Implementation
System requirements:
Operating System: Fedora core 13
Software: Microsoft Visual C++ 6.0 & above, ns 2.34
Hardware: 70 MB space for precompiled ns2 binary or 145 + 50 (nam) MB of space for full ns2 and nam source

NS-2 Simulator
The ns-2 is a network simulator for networking and the results of simulations can be visualized using the Network Animator nam. 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/.

Running the ns
We can start nam with the command 'nam <nam-file>' where <nam-file> is the name of a nam trace file that was
generated by ns, or execute it directly out of the Tcl simulation script for the simulation which we want to visualize. Below we can see a screenshot of a nam window.

Fig. 1. Network animator

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 the network.

Results

We have demonstrated the result of implemented work through various simulations implemented in terms of tcl script like simgridfinal.tcl and simgridmax.tcl

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

ns simgridfinal.tcl

After the execution out.nam file is created inside the current working directory. 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.

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. The following graphs shows the resource usage in terms of the request processed as shown in Graph 1 and time required as shown in Fig.3.

Fig. 3. Nodes versus requests received and processed

Fig.4. Nodes versus utilization time

For a simgridmax.tcl following command is executed on the terminal.

ns simgridfinal.tcl

After the execution out.nam file is created inside the current working directory. 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.
Table II
Comparison of the various requests received and serviced by the nodes

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Number of Requests Received</th>
<th>Number of Requests Serviced</th>
<th>Total time on the node (Seconds)</th>
<th>Actual Resource Utilization (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node2</td>
<td>35</td>
<td>21</td>
<td>4.54</td>
<td>2.59</td>
</tr>
<tr>
<td>Node3</td>
<td>5</td>
<td>5</td>
<td>3.93</td>
<td>0.64</td>
</tr>
<tr>
<td>Node4</td>
<td>10</td>
<td>10</td>
<td>4.54</td>
<td>1.25</td>
</tr>
</tbody>
</table>

Fig. 6 Nodes versus requests received and processed

Fig. 7 Nodes versus utilization time

Conclusion
This paper presents comparative analysis of resource utilization in different peer to peer networks and demonstrates that the number of requests processed at node 2 has been increased with multiple resource consumers and providers.

References