Peer-VM: A Peer-to-Peer Network of Virtual Machines for Grid Computing

(Research Proposal)
Abhishek Agrawal (aagrawal@acis.ufl.edu)

Abstract
This proposal discusses details about Peer-VM which is a peer-to-peer (P2P) networking system allowing users to share the virtual machines (VM) in a P2P fashion for the purpose of Grid Computing. Peer-VM allows for the on demand addition of the virtual machines to the resource pool of existing Grid Computing systems like In-VIGO [9]. This proposal presents the Peer-VM architecture and discusses it’s functionality in detail.

Introduction
In the recent past, two supposedly new approaches to distributed computing have emerged, both claiming to address the problem of organizing large-scale computational societies: peer-to-peer (P2P) and Grid Computing. However one of the basic differences between the two lies in terms of scalability. Grids often involve only small number of participants and are less scalable. One of the reasons for such behavior is that earlier Grid implementations like Globus [10] and Condor [11] did not address scalability and self-management as priorities. On the other hand P2P systems consist of large user communities and are highly scalable. For instance SETI@home [18] consists of several million nodes distributed around the globe. This large scale has emerged from robust self-management of large number of nodes. In this proposal, we discuss Peer-VM which attempts to apply the advantages of scalability and self-management existent in the P2P systems to the Grids.

Recently, interest has been growing towards the use of virtual machines for the purpose of Grid Computing [8]. The In-VIGO system being developed at the University of Florida is a good example of such trend. However, such systems are limited in their capability to scale and add resources on demand. Peer-VM is a peer-to-peer networking system based on which, the Grid Computing systems like In-VIGO can add resources dynamically. The basic concept behind Peer-VM is that a virtual machine can be defined by a set of files which can be easily shared in a peer-to-peer fashion similar to as existent in the present systems like KaZaA [1] or Gnutella [2]. For our purpose, we are using Brunet [3, 6] (a hybrid P2P overlay network being developed at UCLA) as the base P2P system on which Peer-VM is built.
Peer-VM Architecture

Figure 1 below depicts the architecture of the Peer-VM system. At the bottom lies the peer-to-peer routing layer which serves as the backbone of the whole architecture. In our model, we are using Brunet for the purpose of routing. One of the benefits of using Brunet is that it combines the advantages of both the structured (like Tapestry [14], Chord [15], Symphony [16] etc.) and unstructured (like Gnutella, Kazaa, Freenet [13] etc.) subnetworks. We call the nodes on this Peer-VM network as Brunet nodes. Hence, using the unstructured subnetwork, the Brunet nodes would be able to make reliable and scalable global queries and using the structured subnetwork, they will be able to route data to other nodes in an efficient manner.

On top of the P2P routing layer comes the Virtual IP network which is basically the virtual private network of all the Brunet nodes participating in the Peer-VM network. Above the virtual network layer, lies the VM query system which basically serves as the interface to the Resource Manager to search for a particular machine on the Peer-VM network according to the given specifications. On top of the VM query system, lies the resource manager which provides the user with the required application interface.

![Diagram of Peer-VM Architecture](image)

Figure 1: Peer-VM Architecture
Peer-VM Functionality

The actual functioning of the Peer-VM architecture described above has been shown in Figure 2 below. First of all the Resource Manager at peer node 1 gets the request from the user to get a particular virtual machine with the specified parameters. The resource manager forwards this request with the required specifications to the VM query system. On getting the desired parameters, the VM query system performs a search in the Global database and returns the address of the desired virtual machine to the resource manager. After this, the resource manager tries to establish the connection with the remote resource manager (at peer node 2) using the P2P routing layer (Brunet). At this point, the two Brunet nodes needs to hand shake in order to obtain certain information about each other. Such messages are encapsulated in connection control packets which are exchanged in between the two nodes. Once the connection is established, the remote machine on peer node 2 becomes available for use by the user on node 1. After the connection establishment, the actual Ethernet packets that are exchanged between the two nodes are called as Brunet packets. The next section describes the details about the actual communication model taking into account that there can be multiple virtual machines on the same host.
**Peer-VM Communication Model**

Figure 3 below illustrates the Peer-VM communication model. For the purpose of illustration, we are considering user-mode-linux (UML) virtual machines. As shown in the diagram, each host machine represents a particular computer system, such as desktop PC or a server system and may host one or more virtual machines. Each host has exactly one Brunet ID which uniquely identifies that node on the Peer-VM network and each virtual machine on that node has its own private IP address. The virtual machines are connected to the UML switch. One of the UML virtual machine acts as a gateway for all the other machines and interfaces directly with the UML switch. The UML gateway is connected to the tap device which serves as the virtual network interface from where the Brunet daemon can pick up packets and send it over to the actual Ethernet device for transmission.

In the actual scenario, whenever a VM is ready to send a packet to the remote VM on the peer node, it passes that IP packet to the UML switch. The switch determines whether that packet is intended for a machine on the local network or is it for a remote machine running at some other node. If the packet is destined for a local machine, the UML switch simply forwards the packet to the required machine. In case the packet is destined for the remote machine, it passes the packet to the UML gateway which in turn injects the packet into the tap device. The Brunet daemon continuously listens to the tap device and when it sees that a packet is ready for transmission, it picks up the packet from the tap device, appends the Brunet header and forwards the packet to the Ethernet device from where it is sent to the destination via the physical network. The whole process is repeated in the reverse sequence at the destination.

![Figure 3: Peer-VM Communication Model](image-url)
Peer-VM Packet Format

The Peer-VM packet is basically a sandwich of IP, Brunet and IP. It consists of one Brunet header and two IP headers. The packet structure is as shown in the diagram below:

```
<table>
<thead>
<tr>
<th>Ver</th>
<th>IHL</th>
<th>Service Type</th>
<th>Total Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Identification</td>
<td>Flags</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time to Live</td>
<td>Protocol</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Source IP Address</td>
<td>Header Checksum</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Options</td>
<td>Padding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protocol</td>
<td>Hop</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time To Live</td>
<td>Padding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Source Brunet ID</td>
<td>(20 bytes)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Destination Brunet ID</td>
<td>(20 bytes)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IP Header</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Payload</td>
<td></td>
</tr>
</tbody>
</table>
```

Figure 4: Peer-VM Packet Format

Possible Applications of Peer-VM

**Dynamic Resource sharing by Academic Institutions**

One of the possible scenario where Peer-VM system can be used is of a Peer-VM network of academic Institutions where they can share the virtual machines or even some specific applications in a P2P fashion. For instance, if some resource is needed immediately by a Grid Computing system like In-VIGO at University of Florida, it can be added very easily to its resource pool if it is available on a remote machine in the Peer-VM network. Also if a user intends to run an application which is not available in the
home network, then using the Peer-VM system, the resource manager can find a remote virtual machine which is running that application and can provide the user with access to it. The diagram below shows one such possible network of academic Institutions.

Figure 5: A sample Peer-VM Network

**Desktop Distributed Computing**

The concept of Peer-VM can be applied for the purpose of Desktop distributed computing on virtual machines (like Entropia [17]). In this scenario, a user downloads the Peer-VM application from a website and instantiates a virtual machine on his desktop. The address of this virtual machine is returned to the resource manager which then adds the machine to its pool of available resources and can schedule jobs on it.

**References**


