

LAN Machine: A Virtualized Network For Developing & Testing Network & Cloud Applications On A Single Machine

Mohanish Penta¹, Abhishek Prabhudesai², Sunil Bablani³, Veeral Panwala⁴
^{1,4}University of Pune, Maharashtra, India

mohanishp@yahoo.co.in, abhishek.prabhudesai@gmail.com, sun_bablani@yahoo.co.in,
vish_252002@yahoo.com

Abstract – We propose a platform that aims at creating a virtualized network on a single machine. It mainly focuses on network level virtualization. It aims at the consolidation of network level components virtually on a single machine: Wherein user can launch any network-based applications (e.g. - App server's, Web servers, Web services, Internet protocols etc.) on the virtualized nodes having virtual IP's (and all the network layer characteristics) on a single machine. This setup therefore can be used to check the performance issues, related to the applications by simulating various network level scenarios. Hence this platform can be used in the development, testing, learning phase of any network-related application with zero dependency on external network and minimal overhead.

The paper contains solutions delivered for –

Creation of a virtualized LAN environment (effect of LAN) consisting of virtual nodes on a single machine, wherein each virtual node will have its own IP & Subnet.

Launching user application's on the virtualized nodes (in Virtual Network).

And inclusions of Filters which will bring in the simulation of real world scenarios like bandwidth throttling, network traffic, jitters, etc.

So above platform (LAN Machine) setup aims at creating a virtualized Testing platform for testing Cloud related application (i.e. the applications intended to work on the Cloud architectures) on a single machine by creating a simulated real world scenario.

The architecture of the system is discussed for Windows Operating System. But can be extended to UNIX based systems also.

Keywords- Virtual Network Interface Card (VNIC), Service Oriented Architecture (SOA), Virtual Machine (VM)

I. INTRODUCTION

Today virtualization can apply to a range of system layers, including hardware -level, operating system-level virtualization, and high-level virtualization. The commonly used form of virtualization is OS virtualization in which we use VM that virtualizes an OS.

In a study conducted by Global Knowledge [5], Cloud computing is opening new avenues for deploying distributed applications, web services etc. on computer resources hosted by computer providers, the industry is still deficient and is awaiting for the launch of a universal commonly used platform which would manage applications in cloud. The components which are deployed on cloud are app servers, web servers, web-services, SOA applications, etc. Presently the only option of testing the cloud applications is, by hosting them on an actual cloud. This paper discusses an approach for creating a virtualized network infrastructure on a single machine. Using this infrastructure the development and testing of any network based application can be done locally.

II. ISSUES IN EXISTING ARCHITECTURE

Let's have a look on current trends in virtualization in brief –

- Hosted approach.
- Hypervisor based approach.

A **hosted approach** is a virtualization approach where partitioning and virtualization services run on top of a standard operating system (the host). In this approach, the virtualization software relies on the host operating system to provide the services to talk directly to the underlying hardware. VMware's VirtualCenter [12, 13, 14] creates a uniform hardware image implemented in software. With

VirtualCenter, an administrator can manage thousands of Windows NT, Windows 2000, Windows 2003, Linux and Netware from a single point of control.

Issue- With reference to the study conducted by Clear Tube [2, 15], the most significant trade-off of a hosted architecture is in potential I/O performance degradation. Because I/O emulation is done in the host world, a virtual machine executing an I/O intensive workload can accrue extra CPU time switching between the VMM and host worlds, as well as significant time in the host world performing I/O to the native hardware.

In **Hypervisor architecture**, a thin layer of software generally provides virtual partitioning capabilities which run directly on hardware, but underneath higher-level virtualization services. Hyper-V [6] which is a hypervisor-based virtualization system for x64 systems requires an x64-based processor running an x64 version of Windows Server 2008 Standard, Windows Server 2008 Enterprise or Windows Server 2008 Datacenter and a minimum 2GB of memory.

Issue- The problem in this approach is that a high system configuration is required

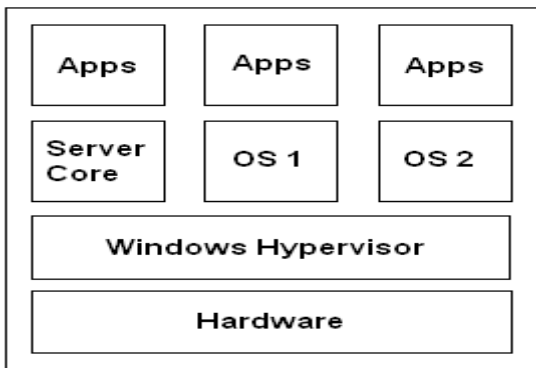


Figure 1 – A simple hypervisor used in Windows

Other commonly used types of virtualization –

- OS streaming
- Application streaming.
- Virtual hosted desktop (VDD).

So the commonly used form of virtualization is OS virtualization in which we use VM to simulate an OS. Let's consider a scenario, a developer is working on a connectivity protocol intended to be used over the web. He wants to check the connectivity and working of the protocol by trying out end to end scenarios across a common platform (e.g. Windows).

Suppose he wants to test the connection between two machines in a network using the new protocol. But when he is out from the company network he only has one machine say his laptop. He can install a VM which will

host another virtual Windows OS for trying out the scenario. But in this case OS virtualization is not actually necessary when the developer wants to test a particular network issue. Like mentioned **there are many scenarios which need only network level virtualization instead of a complete OS virtualization.**

III. ARCHITECTURE

For explaining our methodology, we take a simple example of a chat application. A chat application is a basic example of a network based application. For testing any chat application we need, at least 2 machines connected across the internet or in LAN are. As shown in the figure below.

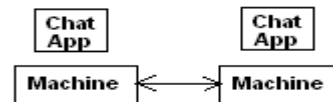


Figure 2 – A Chat Application.

The same can be implemented in **LAN machine** using network level virtualization.

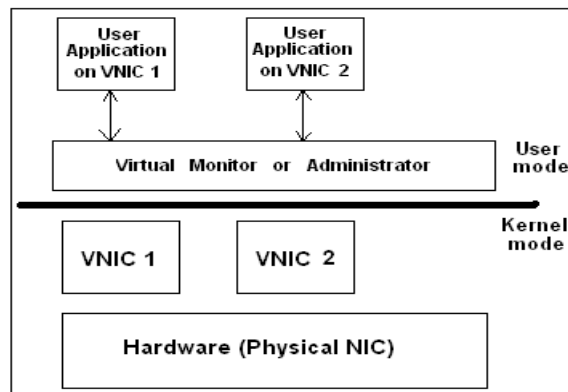


Figure 3 – LAN Machine: HLD 1

Above is a component level diagram which gives a high level view. *LAN Machine* has a virtual monitor which acts like an administrator for controlling the network layer or managing the virtual nodes (VNIC). Administrator is a GUI from where the user will control the system. New virtual NIC's (VNIC) can be added & configured through the administrator. User can configure the VNIC by selecting any IP address along with the subnet & gateway. This GUI will allow the user to launch any application of his choice on any of the virtually configured VNIC's. For eg. In the above figure the "User Application 1" is deployed on VNIC1 & "User Application 2" is deployed on VNIC 2. User can launch more than one application on a VNIC. These applications are now running on different

IP address (Virtual IP's) and can interact locally on a single machine. Thus, in short the architecture will provide the user the freedom to create a virtual network containing any number of nodes.

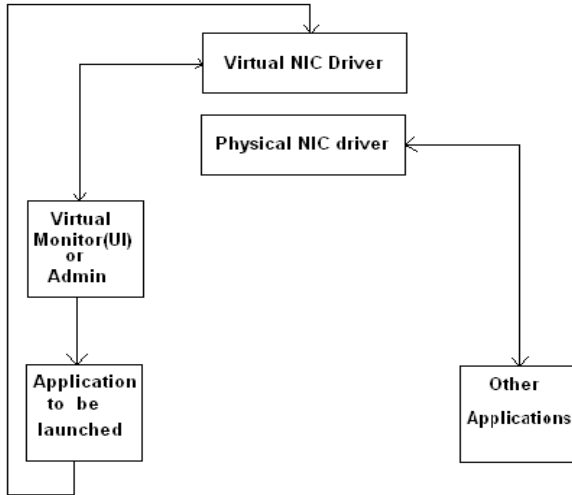


Figure 4 – LAN Machine: HLD 2

Figure above gives a clear idea of how the information will flow through the system. The VNIC driver is placed above the physical NIC. So there will be no interference between virtually generated traffic and the actual traffic. We have divided the implementation of the proposed architecture in 3 parts –

- Virtual Network Interface Card (VNIC).
- Launching a user application through administrator.
- Simulating Network scenarios.
-

A. Implementation of VNIC

In Windows a VNIC can be implemented as a function driver which is similar to an Ethernet device driver. It is a KMDF (Kernel Mode Device driver) Windows device driver. It has the same functionality as that of a physical Ethernet driver. But the VNIC is a connectionless network driver which does not interfere with the underlying NIC hardware or with network to which the physical NIC is connected. Thus the VNIC handles, only the packets which are sent to it through the LAN machine administrator and will not interfere with the packets which the actual NIC handles. Due to this the normal working of the NIC is not disturbed. The VNIC has its own set of ports and mimics the protocol stack of the physical Ethernet driver. The figure below shows a Miniport driver which directly manages a network interface card and provides an interface to higher level drivers. NDIS is short for the

"Network Driver Interface Specification". NDIS also provides a library of functions (sometimes called a "wrapper") that can be used by higher level protocol drivers (such as TCP/IP). The driver explained above thus effectively routes the packets through the apps deployed in the LAN machine administrator, back to the same machine. Each VNIC has a randomly generated MAC address and is registered in the Windows registry.

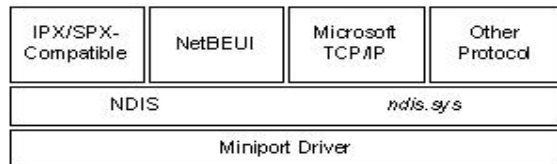


Figure 5 - Connectionless Network Driver Environment.

Figure below shows the exact location where the VNIC function driver will be hooked in the Windows driver stack.

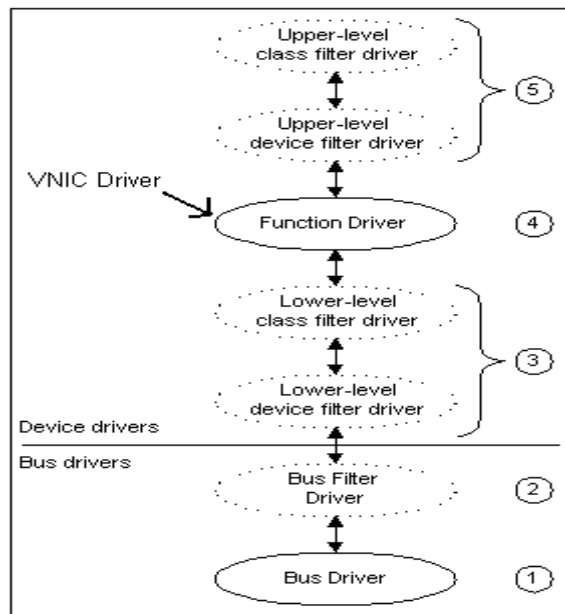


Figure 6- Placement of VNIC driver in Windows driver stack.

B. Launching An User Application through the administrator

The task of launching a user application is achieved by adding a service layer. Figure below shows the traditional approach followed by all web-based applications to establish communication over the network. Winsock is a wrapper to all the lower-level functions for accessing the network card provided by NDIS. In other words, Winsock is an Application programming interface (API) through

which, the behavior of every web-based application which accesses the network can be controlled.

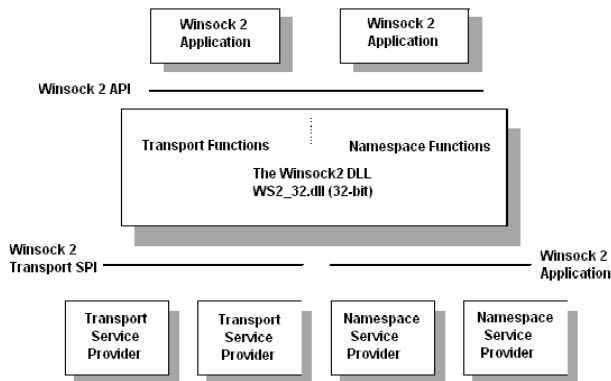


Figure 7 - The Winsock architecture

Whenever a web-based application is launched, it binds and works on the physical IP address of the machine. So when we try launching any web-based application on a VNIC, it by default binds to the physical NIC of the system. This is because Winsock picks up the physical IP address which is maintained in the stack in the Windows kernel. So to change the normal behavior discussed above we have customized the functionalities of the Winsock API. The functionality is changed in such a way that, when the application does a local lookup of finding an IP address, the system returns the list of virtual IP's and not the list of physical IP's. This is only applicable to the applications launched from the GUI or administrator.

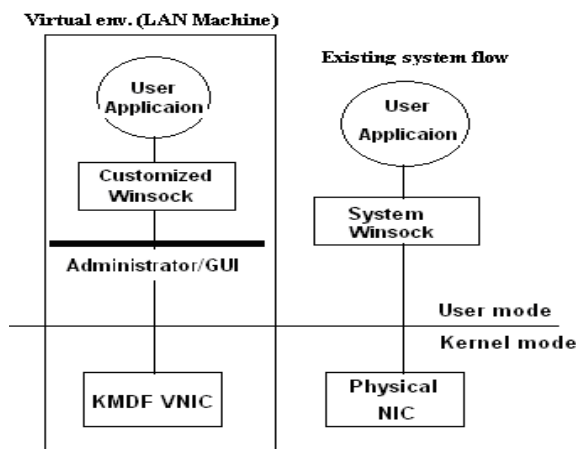


Figure 8 - Functioning of customized Winsock

It will not interfere with the other network related requests on the machine.

C. Monitoring Network Characteristics

After the user application is deployed in the virtual environment, the user can then generate virtual traffic and monitor it. Various network scenarios can be generated using this platform artificially. This functionality can be implemented using various approaches. (User can use any of the custom approach as per the network scenario to be simulated) Some suggested one's are –

Approach I – Creating a Filter Hook.

Approach II – Windows Filtering Platform.

Approach I –

A filter-hook driver is a kernel-mode driver that implements a callback function called a filter hook and registers that callback function with the system-supplied IP filter driver. The IP filter driver then uses the filter hook to determine how to process incoming and outgoing packets. To supplement the Packet Filtering application programming interface (API), a filter-hook driver can be created to filter network packets. User-mode applications use this API to create and manage, in the system-supplied IP filter driver, input and output filters that filter packets with specific IP addresses or TCP/UDP port numbers. Because the Packet Filtering API optimizes the system-supplied IP filter driver to process packets without the overhead that is associated with a filter-hook driver.

A filter hook performs the following actions on incoming and outgoing packets-

- PF_FORWARD
- PF_DROP
- PF_PASS

Based on the functionality provided by the filter hook driver we can implement the different kind of network monitoring scenarios.

Approach II –

Windows Filtering Platform (WFP) is architecture in Windows Vista and Windows Server 2008 that filter's and modify TCP/IP packets, monitor or authorize connections, filter Internet Protocol security (IPsec)-protected traffic, and filter remote procedure calls (RPCs). WFP provides APIs so that you can participate in the filtering decisions that occur at several layers in the TCP/IP protocol stack. Using WFP custom scenarios can be generated for monitoring the network traffic, testing the performance, intercepting the network traffic etc.

Above given are two approaches of achieving network monitoring in the virtualized networks. Hence custom scenarios can be added by the user as per his need. There are many other ways in which this can be achieved.

IV. EXAMPLE: USE IN SOA

Suppose there is a system, which remotely access a JAVA component which is running over some machine in the network & even access a C++ component which is running over some different machine in the network in loosely coupled SOA architecture. So when a developer is writing such an application he is dependant on at least 3 machines which are connected in the network. This can also be achieved by using VM's. But when testing for network performance issues VM's lack efficiency and incur large CPU overheads.

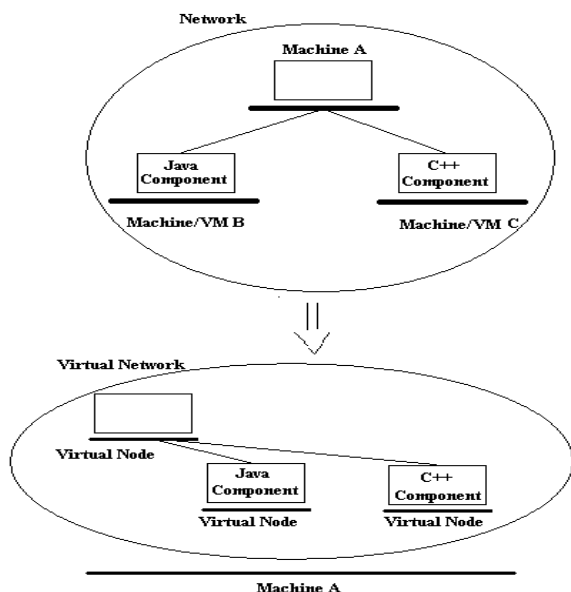


Figure 9 - Use of LAN Machine in SOA

This scenario can be efficiently simulated using the *LAN machine*. Figure above shows two scenarios one using the architecture discussed and one without this architecture. The virtual nodes in the second part indicate virtualized components which virtually exist on the same machine. These virtual nodes have their own independent IP address, Subnet & gateway. In brief the idea is to simulate LAN environment on a single machine i.e. these virtual nodes depict physical machines which are connected in LAN. This in a way forms a virtual network on a single machine.

V. FUTURE WORK

The current architecture of LAN Machine simulates a network, user applications can be then deployed on the virtual NIC's. The system can be ramped up further by adding various upper level filter's in form filter drivers. These filters will simulate the different network scenarios like load, network traffic, bandwidth throttling, jitters, etc.

So by adding the necessary functionality in the GUI or the administrator this architecture can also be used for testing the performance of various web-based applications. It will help the developer to find the transactions per second (TPS) rate that an application can support when use in a network.

This VNIC architecture can be used for launching hypervisor enabled VM's on a single machine. A VM can be launched on physical NIC of the machine as well on the virtual NIC of the machine. This architecture (platform) can be further extended to virtualization of switches & routers.

VI. CONCLUSION

The architecture presented in this paper describes a new way of achieving network virtualization with minimal performance overhead. This architecture can be effectively used in varied fields for achieving high efficiency and productivity. It aims at increasing the productivity in the development phase of Software Development Life Cycle (SDLC).

VII. REFERENCES

- [1] Virtualization Performance Study by Clear Tube.
- [2] Developing an Enterprise Client Virtualization Strategy – By John Dunlop, Intel Corp.
- [3] IEEE. 802.1ak multiple VLAN registration protocol.
- [4] Comparing Two & Four Socket Platforms For Server Virtualization – By Sudip Chahal, Intel corp.
- [5] Virtualization Evolution to Revolution –By CDW
- [6] Hypervisor Functional Specification – By Intel Corp.
- [7] Cloud Architectures – Jinesh Varia , Amazon Web services.
- [8] Networking & Unified Communication – By CDW.
- [9] Virtualization Overview – By Global Knowledge.
- [10] Understanding Performance in Virtualized Environment by Krishna Raj Raja
- [11] Performance Trends by Richard McDougall, Chief Performance Architect, VMware, Inc.
- [12] ESX Server architecture and performance implications
http://www.vmware.com/pdf/esx2_performance_implications.pdf
- [13] ESX Server performance and resource management for CPU-intensive workloads
http://www.vmware.com/pdf/esx2_performance_implications.pdf
- [14] Advanced Best practices Using VMware Virtual SMP
http://www.vmware.com/pdf/vsmp_best_practices.pdf
- [15] Performance tuning best practices for ESX server 3
http://www.vmware.com/pdf/vi_performance_tuning.pdf