# DEVELOPMENT OF A LAN SIMULATION TOOL BASED ON WINDOWS ENVIRONMENT

Hamid M. Ali, Nidhal Ezzat and Wisam F. Kadhim
University of Baghdad

**ABSTRACT**
The Internet's rapid growth has spurred the development of new protocols and algorithms to meet changing operational requirements such as security, multicast transport, mobile networking, policy management, and quality of service support. Development and evaluation of these operational tools requires answering many design questions. This work proposes a computer network simulation program, devoted for wired LAN systems. The simulator would be able to work under Microsoft Windows NT platforms, also it has the potential to provide an emulation environment which should be suitable for testing protocols above the TCP layer under the Windows NT platform supported network layers, and offering scalability by running the simulator under distributed network system.

**KEYWORDS**
**Network Simulators, Modeling.**

**الخلاصة**
أدى التنامي السريع للإنترنيت إلى تشابك عملية التطوير لبروتوكولات وخوارزميات جديدة لتحقق المتطلبات العملية المتغيرة، مثل أمن المعلومات (Security)، شبكات الأجهزة المتنقلة (Mobile Networks)، إدارة اسلوب العمل (Policy Management)، جودة الخدمة (Quality of Service). تطوير وتقييم هذه الأدوات التشغيلية تتطلب الإجابة عن العديد من التساؤلات التصميمية. هذا البحث يقترح نظام محاكاة لشبكات الحاسبات، مخصص لمنظومات الـ(LAN) السلكية. نظام المحاكاة المقترح يكون قادر على العمل ضمن نظام التشغيل (Microsoft Windows NT)، كذلك له القابلية على توفير محيط والذي يكون مناسباً لفحص البروتوكولات ما فوق (TCP layer) باستخدام طبقات الشبكات (network layers) التي توفرها أنظمة التشغيل (Windows NT platform)، و يوفر نظام المحاكاة المقترح المقياسية (Scalability) عن طريق تشغيل المحاكاة في أنظمة شبكات موزعة (Distributed Network System).

## INTRODUCTION
The network simulators exist in many forms and work on different criterion of computer networks, some of these simulators can combine the simulation of more than one network system, and others can be specifically created to simulate a particular network protocol. This fact has brought into the surface some drawbacks in the simulation level of the current simulators, in reference to the concept of the common simulator [BEF00] that can work on most of the network systems and protocols.

The upside of this divergence in functionality is realistic marketing; some vendors' simulators are optimized for managing LANs (Local Area Networks) and others for WANs (Wide Area Networks), some merely provide network diagramming and limited simulation while others perform more sophisticated global network modeling. The downside is that no one simulator can perform all the required functionality, where if a particular network is needed to be modeled, analyzed, and simulated, then multiple simulators are required, besides the notable differences between them that imply the same job [Jim98].

Most of the simulators can simulate all of the network elements, but some of the packages fall short. Where some can't simulate disks, chips or controllers, some can't mirror frame queuing performance or media speed, and others can't simulate device-level details such as hardware architectures. With the exception of some simulators, most of the simulators are not considered to be system-level simulation devices. That means they are limited in their ability to determine how the performance of the end stations will impact the performance of the network [Jim98].

Another important issue to consider is that most of the network simulators do not support the emulation feature; that is linking the simulated network to an actual real network, thus extending the study criterion on the performance and behavior.

## - OBJECTIVE OF THE WORK

In this work, the modeling of the basic network components used (clients, servers, hubs, and routers) was accomplished using Windows NT resources and the proposed network simulator had a simulation engine capable of: adding and removing of network components during runtime, configuring network components' parameters, and performance results calculation. The proposed network simulator also has a timed events management to perform the required simulation scenarios, like: server crash, router failure, or link sudden disconnection, where these events are user generated according to specified time during the simulation run.

The work included a validation test to the proposed network simulator by running a file transfer application (as an example) on a simulated network from one hand and a similar real network on the other hand all under the same timed events, then comparing the simulation results.

## - MODELING OF NETWORK ELEMENTS

The network resources of Windows NT Platform were used to model each network element, where they were based on some of the network architecture functionality of Windows. The network protocols of Windows (like TCP/IP) were used as the protocol layers in the simulation engine, instead of programming prototypes of these protocols (like other simulators do, especially those programmed under UNIX), this would make the protocol modeling approaches much more to reality, since the actual protocol would be used in the simulation process.

The modeling of end-systems, like clients and servers, were based on creating a network resource by adding virtual network adapters, so that Windows network architecture would assign a set of protocol layers on top of each virtual network adapter, which would lead to an accessible network resource representing either a client or a server.

The modeling of the intermediate-systems, like hubs and routers, were based on assigning connections' lists and routing tables that would be used in packet filters within the protocol layers in the Windows network architecture, so that each connection's list would represent a hub, and each routing table would represent a router.

## - WINDOWS NT NETWORK ARCHITECTURE

Microsoft® Windows® 2000 and later operating systems use a network architecture based on the seven-layer networking model developed by the ISO (International Standards Organization). Introduced in 1978, the ISO OSI (Open Systems Interconnection) Reference model describes networking as: "a series of protocol layers with a specific set of functions allocated to each layer. Each layer offers specific services to higher layers while shielding these layers from the details of how the services are implemented. A well-defined interface between each pair of adjacent layers defines the services offered by the lower layer to the higher one and how those services are accessed" [Mic01a]

Figure 1 represents a model of Windows 2000 network architecture. In the figure, components that are on the same horizontal level provide similar functionality. The top layer of the diagram is where user applications reside. In order to communicate with other networked computers, additional software and hardware support is needed. Each layer below the applications and services layer provides services that are necessary to create packets of data, arrange for their delivery, and send them across the physical media to another computer.

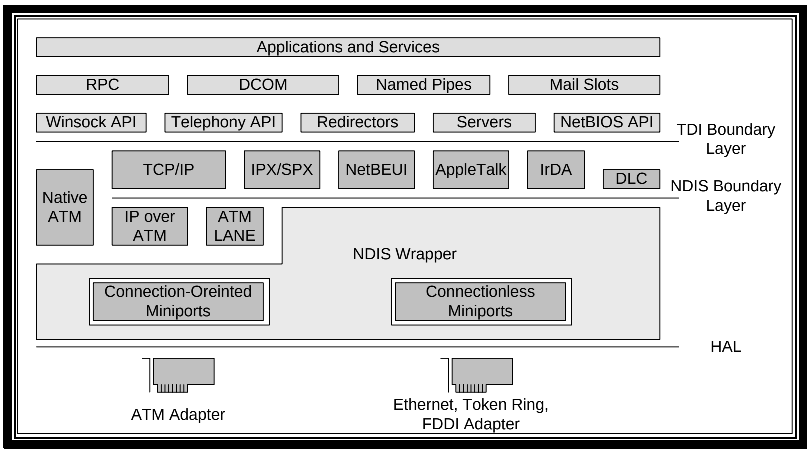

**Figure 1:** Windows 2000 Network Architecture Layers [Mic01a].

The Windows Network Architecture provides packet filtering techniques that can be used to filter and process inbound and outbound TCP/IP transport protocol data. There are two types of these techniques: user mode techniques and kernel mode techniques. Figure 2 shows some of these techniques.

User mode is the processor mode in which applications run. User mode processes have access only to their own address space and must use established interfaces to obtain other system services. This protects the operating system and improves its performance [Mic01b].

Kernel mode is the processor mode, in which operating system code (such as system services and device drivers) runs. It is a highly privileged mode of operation where the code has direct access to all memory, including the address spaces of all user-mode processes and applications, and to hardware. Processes running in kernel mode have access to advanced CPU features for I/O and memory management [Mic01b].

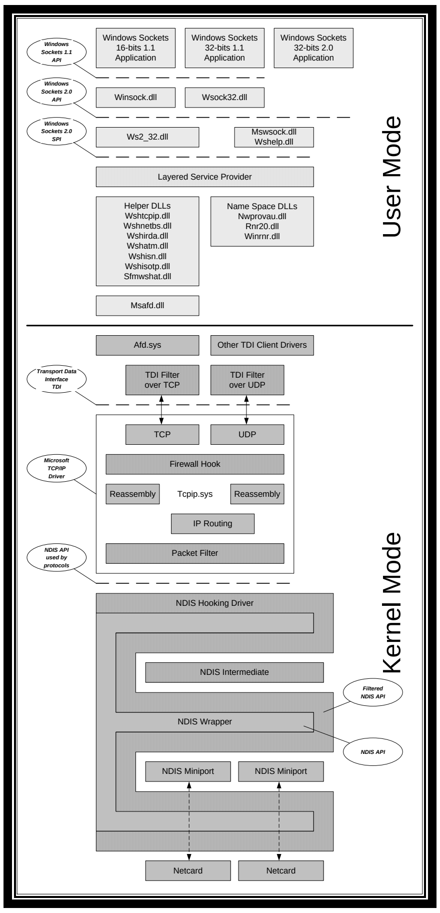

**Figure 2:** Some TCP/IP Traffic Processing Techniques[Mic01b].

## - PROPOSED NETWORK SIMULATOR

In order to meet some of the intended simulation requirement of the network research [BHH99], this work focused on the design of a network simulator based on the following specification:

- Simulation criterion that focuses on the impact of changes in network behavior on the application layer and user mode applications.
- Use of Microsoft® Windows® NT Platform to model the network elements, where Windows network resources are used to model the targeted network elements, like clients, servers, hubs, and routers.
- The network simulator models and functions would be included as DLLs (Dynamic Linked Libraries), so it can be used under any supported platform and with wide variety of network applications, thus leading to increase in level of abstraction.
- Use of flexible graphical user interface that would improve the interaction with the network simulator, and make easy access to its functions.
- Applying a simulation scenario based on network components' failures and recovers on specified times by the user to simulated real world events on networks, like router failure or server crash.
- Provide the potential for supporting full emulation capabilities, where the simulated model can be accessed directly form an outside real network and vice versa, so that all the traffic can be monitored and data analyzed.
- Capability of simulating large scale networks depending on the full emulation potential support, using parallel operation of multiple simulators running on different machines connected via a real network, thus reducing both time and resources.

## Simulator Design

### *A. Clients & Servers Models*

In order to model the clients and servers, it should be pointed out that each client and each server represents a system of applications running on a networking protocols stack that are on top of one NIC at minimum. Therefore, to create a model of a client (or a server), an NIC should be added to the system, so that the Windows device management would add the appropriate protocol layers and drivers to make this NIC functional and the applications to work on it.

Since it's not convenient to add physical NICs to the system to represent the clients and servers, another method can be applied instead, which is through the use of virtual NICs. These virtual NICs are merely driver entries at the miniport diver layer or intermediate miniport driver layer, where there is no actual NIC bound to it. Windows network architecture provides two types of these virtual NIC drivers: Microsoft Loopback Adapter, and NDIS MUX Intermediate driver.

There is no documentation about the Microsoft loopback adapter, except the installation method and the general use of it. Microsoft provided this loopback adapter as a dummy NIC that can be used for troubleshooting the network configurations and settings, another advantage of Microsoft loopback adapter is achieved through installing Active Directory on a Domain Controller that doesn't have a network adapter. Normally, this is not a possibility as active directory requires that a network adapter be installed first. If it is needed to install active directory on a computer without a network card, such as a test or a lab environment, Microsoft loopback adapter can be installed instead of physically installing a network adapter.

The MUX intermediate miniport driver is an NDIS 5 driver that demonstrates the operation of an N:1 or 1:N MUX driver, i.e. one which creates multiple virtual network devices on top of a single lower adapter. Protocols bind to these virtual adapters as if they are real adapters [Mic01a]. The number of virtual miniports exposed by a MUX intermediate driver can be different than the number of lower physical adapters that are bound to the driver. A MUX intermediate driver exposes virtual miniports in a 1:N, N:1, or even an M:N relationship with underlying adapters. This results in complicated internal bindings and data paths [Mic01a].

In an N:1 configuration, a MUX intermediate driver can expose many virtual miniports for a single physical adapter below. Overlying protocols bind to these virtual miniports of the MUX intermediate driver in the same way that they bind to non-virtual miniports. The MUX intermediate driver handles requests and sends that are submitted to the driver for specific connections at each virtual miniport. The driver repackages and transfers these requests and sends down to the NDIS miniport driver for the bound physical adapter. Figure 3 illustrates an N:1 MUX intermediate driver configuration [Mic01a].

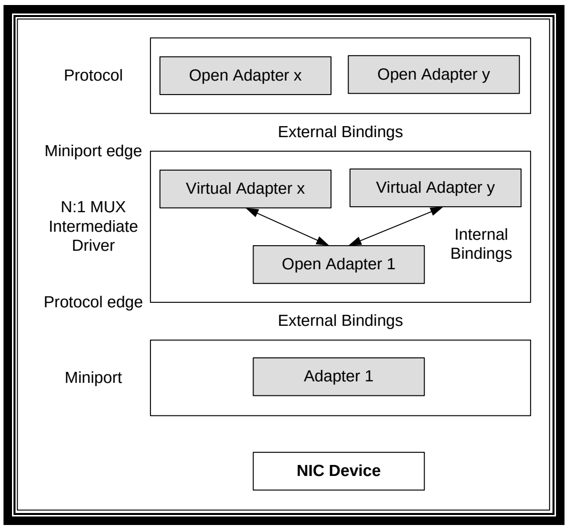

**Figure 3:** NDIS N:1 MUX intermediate driver configuration[Mic01a].

*B. Hubs & Routers Models*

Since there are no real NICs, there exist no actual physical connection between the simulated clients and servers; also it would mean there is no physical routing operation can be applied. So in order to model the physical connections, hubs and routing operation, it would be useful to use the packet filtering techniques provided by Windows network architecture.

The most useful technique to use for modeling the physical connections, hubs, and routers is the Firewall-Hook driver. This fact comes from the following aspects (refer to figure 2 for details):

- The Windows network architecture TCP/IP routes all the internal traffic at the protocol driver layer (TCP/IP driver), which means there no traffic is going down to the NDIS layers below.
- All the configured network adapters that exist in the system, whether they were actual or virtual, are treated as being all connected together, even if they have different IP addresses.
- The firewall-hook driver is located within the TCP/IP driver on top of the IP routing functions.
- It's a kernel mode packet filter that performs its operation on all the packets traffic of every application running on the system. Yet, it supports the interaction of the user on its operation, which makes it user mode configurable.

Therefore, the modeling of the physical connections, hubs, and routers can be made through modeling only the lists of connections and routing tables. And theses lists and tables are passed to the firewall-hook driver to perform the connections and routing operation, provided that the routing is only static.

The filter hook driver only allows one filter function installed in the system. If one application already uses this functionality, the targeted application doesn't work. With firewall-hook driver, this problem has no existence, all filter functions can be installed as required. Each filter function has a priority assigned, so the system will call one function after another (in priority order) until a function returns "DROP PACKET". If all functions return "ALLOW PACKET", the packet will be allowed. This can be thought as a chain of functions, this chain is broken when one of them returns "DROP PACKET", and the order of each function in the chain is given by its priority value [Jes04].

Figure 4 represents an example of the firewall-hook driver operation, with the following processes:

- A packet is received at the host, IP driver has the list of filter functions ordered by priority (the function with more priority is Filter Function 1).
- First, the IP driver passes the packet to the highest priority filter function and waits for the return value.
- Filter Function 1 returns "ALLOW PACKET".
- Because Filter Function 1 allows the packet, IP driver passes the packets to the next filter function: Filter Function 2.
- In this case, Filter Function 2 returns "DROP PACKET". So, IP driver drops the packet and does not continue calling next filter function.
-

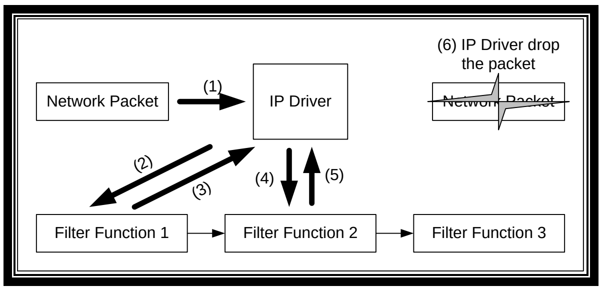

**Figure 4:** An Example of Firewall-hook Driver Operation[Jes04].

Another problem that can be found with filter hook drivers is that for sent packets, the user can't access packet content data. However, all data can be accessed with a firewall-hook driver. The structure of data received in a firewall-hook filter function is more complex that the one received in filter hook driver. It's more similar to the structure of packets that can be found in an NDIS driver, where the total packet is composed by a chain of buffers. As filter hook driver, firewall-hook driver is only a kernel mode driver used to install a callback function (but firewall-hook driver installs a callback in IP driver). In fact, the process to install a firewall-hook driver is similar to the one used to install a filter hook driver [Jes04].

### *C. Simulation Engine*

The simulator engine consists of three major functions: adding the modeled network components (Clients, Servers, Hubs, and Routers) to the system during runtime, configuration of network parameters of each installed component, and calculating the performance results of the simulated network. The design of the simulator engine would also consider the support for full emulation, such that the simulator would be beneficial at all levels of usage.

After the modeled network components are added in the system as designed to form the simulated network, it's required to configure the network parameters of each added component, like IP address, subnet mask, default gateway, and DNS (Domain Name Service) server. Microsoft Windows provides a set of classes and functions that can be used for this purpose, these classes and functions are found in WMI (Windows Management Instrumentation).

WMI is a component of the Microsoft Windows operating system and is the Microsoft implementation if the WBEM (Web-Based Enterprise Management), which is an industry initiative to develop a standard technology for accessing management information in an enterprise environment. WMI uses the CIM (Common Information Model) industry standard to represent systems, applications, networks, devices, and other managed components. WMI can be used to automate administrative tasks in an enterprise environment [Mic01a].

When activating the designed network in simulation mode, it's important to calculate the performance of the network, like bandwidth, percentage of sent and received packets, and each protocol status. The mechanism by which Windows collects performance data on various system resources is the *performance counter*. Windows contains a pre-defined set of performance counters with which the user can interact; some of these are found on all Windows 2000 systems and some are custom to specific applications and are found only on certain systems [Mic01a].

### D. Events & Time Management

The events and time management in the proposed network simulator would include user generated scenarios that would run in a timed-event sequence. The user would specify the type of action to be taken and the corresponding time for the action to occur. These actions can take the form of servers' crashes, routers failures, and links sudden disconnections, so that the user can measure the degree of interaction between the network nodes and explore the vulnerable points in order to avoid them.

This type of management can be achieved through the use of a time ordered list or table that would be referenced on a time bases in respect to a simulation timer, where at every time tick the table would be referenced to check whether the current simulation time matches one of the events time. At that point, if a match found, the events and time manager would execute the event using WMI functions to disable a certain virtual adapter that models a client or a server, or remove a set of filtering rules that represent a hub or a router.

Each events table entry would contain the following information to represent an action:

- Time of the event, which is referenced to the simulator timer.
- Network node to be disabled, that indicates whether a client, server, hub, or a router to be crashed.
- Duration of the action, where it sets the time for which the action would take from failure to recovery.

## System Architecture

The simulator system architecture is divided into two main sections (or layers): Simulation Platform (VNetSimPlatform), and Application Layer (VNetSimApplication). The reasons behind such architecture are:

- Provide transparency of the kernel mode used resources from the application layer that is used by the user.
- Make the simulation platform as stand alone set of resources, which makes it easier for future development to upgrade the functions within the platform or the application layer without affecting the other.

Figure 5, shows the general system architecture of the network simulator.

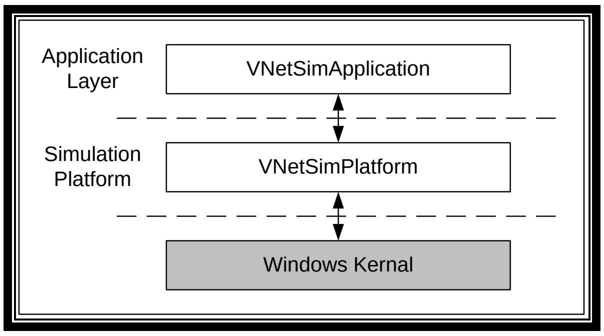

**Figure 5:** Simulator General System Architecture.

*E. Simulation Platform (VNetSimPlatform)*

This layer is considered as the kernel mode layer of the simulator, it provides the functions that use windows resources to perform the modeling of each network component (such as Clients, Servers, Routers, and Hubs) and manage these resources to appear as objects to the upper application layer. The simulation platform consists of a layered set of DLLs (Dynamic Linked Libraries); each one handles certain functions or models a set of network components. Figure 6 shows the components of the simulation platform.

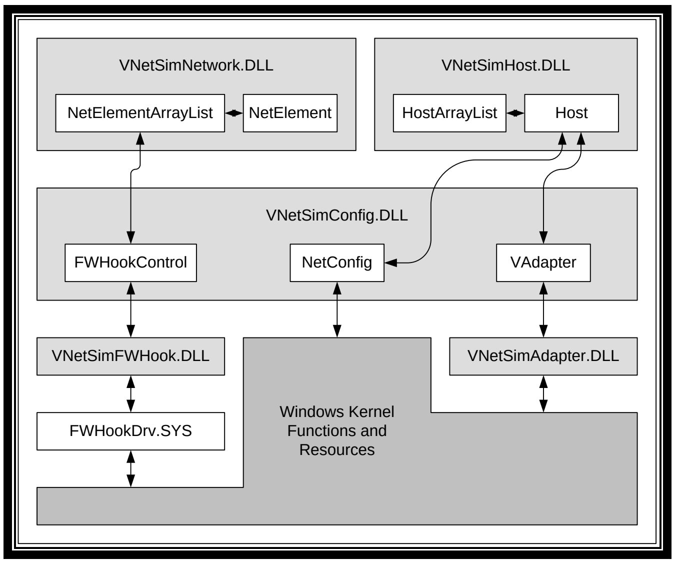

**Figure 6:** Components of Simulation Platform.

The simulation platform uses a collection of programming languages to perform its functions, where at the lower layers conventional C language was used with drivers (such as FWHookDrv.SYS), also the C++ language was used with the DLLs that operate with Windows system calls and functions (such as those used in VNetSimAdapter.DLL and VNetSimFWHook.DLL). At the upper layers, C# language was used with the DLLs that work as objects of the modeled network components (such as VNetSimHost.DLL and VNetSimNetwork.DLL), as well as the DLLs working as interfaces between the object DLLs and the kernel mode DLLs (such as VNetSimConfig.DLL).

The reason for such variety in programming languages is that at every level of the simulation platform there are certain needs that each programming language can provide better than the others. In the case of drives, commonly conventional C language can be used in order to compile and build the driver. As for the DLLs that operate on Windows resources and kernel functions, the C++ language is known to be a very efficient and strong tool to work with Windows resources and supports their requirements from pointers, complex data structures, etc. Whereas C# language support for pointers and complex data structures is considered to be unsafe code by the complier.

The C# language is easier to deal with than C++ in regard to the reduce in complexity and the compilation features provided by the Dot Net environment, also in aspects of developing the graphical user interface, which makes the C# more convenient to use at upper layers that is used by the application layer as well as to hide the complexity of the underlying functions and resources [TWA02].

*F. Application Layer (VNetSimApplication)*

The application layer performs most of the management functions and operations performed by the network simulator, where it can be considered as the simulator kernel. The application layer consists of four main blocks of functions, as shown in figure 7, each block is responsible for certain set of functions.

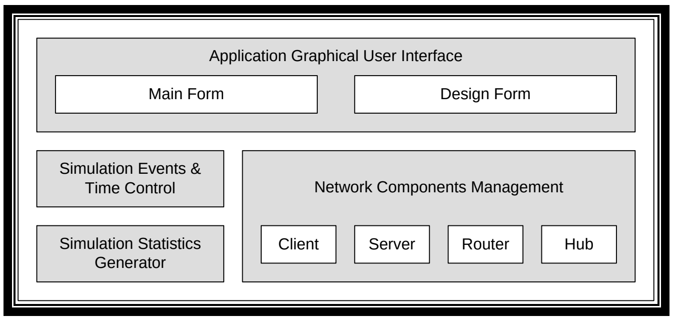
Application Graphical User Interface
Main Form
Design Form
Simulation Events & Time Control
Network Components Management
Simulation Statistics Generator
Client
Server
Router
Hub

**Figure 7:** Block Diagram of the Application Layer.

• Application Graphical User Interface

This block represents the graphical user interface forms that would appear to the user. It handles the operations and calling functions associated with each graphical object, such as menus and toolbars, and pass the control to the corresponding block to perform the required functions. The application graphical user interface contains two classes: main form class, and design form class.

The main form class is a windows form class that represents the main window of the graphical user interface. It holds the main menu, the main toolbar and it works as a container for the design form. The design from class is also a windows form class that works as the document where the users draw the design of the network to be simulated by using network components icons found in the main form toolbar.

• Network Components Management

This block holds the graphical and functional classes that represent the network components, such as Clients, Servers, Hubs, and Routers. The main function of the network components management block is to provide a transformation between the application layer space objects into the simulation platform space objects so as to keep control over the array lists of objects at the two spaces.

For example, if the user adds a client to the design form, a client object is added to the application layer space; hence, it is added to the clients' array list along with its configuration parameters. On the other hand, the network components management block would make a transformation of the client object from the application layer space into the simulation platform space by adding a host object to it, so it is added also to the hosts' array list along with the client's configuration parameters. This step would add a loopback adapter to the system to represent that client object and the adapter's configuration is set according to the network parameters of the client object.

• Simulation Events & Time Control
This block contains a single class which is called Simulation class. It contains a timer object to provide the necessary timing for the simulation runtime process. The class also contains an array list that works as a timed-events table, so it can hold the type of events to be activated during the simulation run in a timely manner.

• Simulation Statistics Generator
This block is responsible for generating the statistics report of the simulation run, where it contains instances of the performance counters built in the firewall hook driver, which would provide the data required by the user to analyze the performance of the simulated network under specific events.

## * IMPLEMENTATION AND PERFORMANCE

The network simulator was implemented mostly in Microsoft .Net C# language, except for some parts of the simulation platform, where it can operate on Windows NT platforms (Windows 2000 Server, Windows 2000 Professional, Windows XP Home Edition, Windows XP Professional Edition, and Windows 2003 Server).

The network simulator was deployed under Pentium 4 with Windows XP SP2 and a validation test was implemented on the proposed network simulator by running a file transfer application (as an example) on a simulated network (shown in Figure 8) from one hand and a similar real network on the other hand all under the same timed events listed below:

- Client 1 performs a download operation from the file server at early start of the simulation process, during that operation the file server crashes for 2 seconds then recovers.
- After several seconds, Client 3 performs another download operation from the file server, during that operation the network hub experience a power failure for 5 seconds then recovers.
- Client 2 performs an upload operation to the file server to update some data, during that operation client 2 experiences a system halt for 5 seconds then recovers.
- Finally, client 3 performs a download operation from the file server, but this time client 3 experiences a system halt for 5 seconds then recovers.

By comparing the simulation results, the validation test results (Figure 9) showed an acceptable tolerance with slight difference that comes form the several factors like: the variable processing conditions of the operating systems where the application was running on, the tolerance of simulation and modeling of the proposed network, and accuracy of timed events when applied at both simulated and real networks.

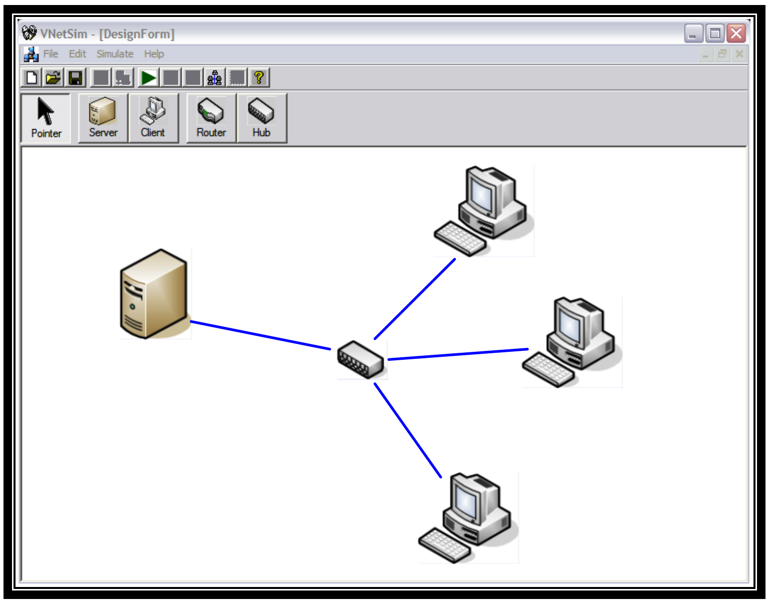

**Figure 8:** Network Topology that the Sample Application would be tested on.

| From | | To | | Simulated Bytes/sec | | Real Bytes/Sec | |
|---|---|---|---|---|---|---|---|
| Device | IP Address | Device | IP Address | Sent | Received | Sent | Received |
| Client 1 | 192.168.10.1 | Server | 192.168.10.10 | 302 | 1512 | 300 | 1550 |
| Client 1 | 192.168.10.1 | Server | 192.168.10.10 | 231 | 1345 | 250 | 1350 |
| Client 1 | 192.168.10.1 | Server | 192.168.10.10 | 133 | 0 | 150 | 3 |
| Client 1 | 192.168.10.1 | Server | 192.168.10.10 | 213 | 342 | 225 | 350 |
| Client 3 | 192.168.10.3 | Server | 192.168.10.10 | 312 | 383 | 315 | 300 |
| Client 3 | 192.168.10.3 | Server | 192.168.10.10 | 462 | 1234 | 460 | 1300 |
| Client 3 | 192.168.10.3 | Server | 192.168.10.10 | 12 | 0 | 15 | 7 |
| Client 3 | 192.168.10.3 | Server | 192.168.10.10 | 21 | 23 | 30 | 30 |
| Client 2 | 192.168.10.2 | Server | 192.168.10.10 | 123 | 462 | 130 | 475 |
| Client 2 | 192.168.10.2 | Server | 192.168.10.10 | 1351 | 235 | 1400 | 235 |
| Client 2 | 192.168.10.2 | Server | 192.168.10.10 | 0 | 125 | 5 | 130 |
| Client 2 | 192.168.10.2 | Server | 192.168.10.10 | 12 | 152 | 15 | 155 |
| Client 3 | 192.168.10.3 | Server | 192.168.10.10 | 342 | 351 | 350 | 350 |
| Client 3 | 192.168.10.3 | Server | 192.168.10.10 | 43 | 1523 | 50 | 1550 |
| Client 3 | 192.168.10.3 | Server | 192.168.10.10 | 9 | 231 | 10 | 235 |
| Client 3 | 192.168.10.3 | Server | 192.168.10.10 | 12 | 521 | 15 | 525 |

**Figure 9:** Network Statistics resulted from running the Sample Application.

## * CONCLUSION AND FUTURE WORK

The network resources of Windows NT platform can be used to model each network element, where they can be based on some of the network architecture functionality of Windows.

The modeling of end-systems, like clients and servers, can be based on creating a network resource by adding virtual network adapters, so that Windows network architecture would assign a set of protocol layers on top of each virtual network adapter, which would lead to an accessible network resource representing either a client or a server.

The modeling of intermediate-systems, like hubs and routers, can be based on assigning connections' lists and routing tables that would be used in packet filters within the protocol layers in the Windows network architecture, so that each connection's list would represent a hub, and each routing table would represent a router.

It was found that using timed events table for events management is the best way to make user defined simulation events, where the user is free to set any type of events to occur at a specific time during simulation run, these types of events can take the form of: server crash, router failure, or link sudden disconnection. Those user-generated events would help in measuring the degree of interaction between network nodes, and explore the vulnerable points in network application development in order to avoid them.

The proposed network simulator has the potential to continue the research in the following topics of network simulation:

- Use of Mux intermediate driver to model the end-system network components instead of Microsoft loopback adapter, and modify the driver source code to support more user interaction in its operation, like embedding the routing and hub connections within the Mux driver.
- Develop new virtual drivers that would model wireless LANs, and ATM NICs, that could be used to expand the criterion of protocols the network simulator would support and simulate.
- Develop the application layer of the simulator to support more functionality regarding the scenario generation of events and time management functions, also to provide sniffing and more advanced performance counters and measures to be useful in the analysis of the simulated network.

## REFERENCES

- [BEF00] Lee Breslau, Deborah Estrin, Kevin Fall, and Sally Floyed, *"Advances in Network Simulation"*, IEEE Computer, Vol. 33, No. 5, p. 5967, http://citeseer.ist.psu.edu/bresl au00advances.htm, 2000.

- [BHH99] Sandeep Bajaj, Padma Haldar, Mark Handley, and Ahmed Helmy, *"Improving Simulation for Network Research"*, Technical Report 99-702, University of Southern California, Los Angeles, http://citeseer.ist.psu.edu/bajaj99improving.html, March 1999.

- [Jes04] Jesús O., *"An Adventure: How to Implement a Firewall-Hook Driver"*, Code Project, Article available at: http://www.codeproject.com/internet/FwHookDrv.asp?df=100&forumid=121826&exp=0&select= 960283, 2004.

- [Jim98] Duffy Jim, *"Simulation Tools are as Varied as LANS They Model"*, Article available at: http://www.findarticles.com/p/articles/mi_qa3649/is_199802/ai_n8790436.html, 1998.

- [Mic01a] Microsoft Corporation, Microsoft Windows XP Driver Development Kit, *"Network Drivers"*, 2001.

- [Mic01b] Microsoft Corporation, Microsoft Development Network, Windows NT Workstation 4.0 Resource Kit, *"Kernel Mode and User Mode"*, April 2001.

- [TWA02] Adrian Turtschi, Jason Werry, Joseph Albahari, and Greg Hack, *"C#.Net Web Developer's Guide"*, Syngress Publishing, Inc., 2002.

**ABBREVIATIONS**

| | |
|---|---|
| ATM | Asynchronous Transfer Mode |
| CIM | Common Information Model |
| DLL | Dynamic Linked Library |
| DNS | Domain Name Service |
| ISO | International Standards Organization |
| LAN | Local Area Networks |
| NDIS | Network Driver Interface Specification |
| NIC | Network Interface Card |
| OSI | Open Systems Interconnection |
| TCP/IP | Transmission Control Protocol/Internetwork Protocol |
| WAN | Wide Area Networks |
| WBEM | Web-Based Enterprise Management |
| WMI | Windows Management Instrumentation |