Network Protocol Development Suite

By

Brook Patten

Submitted to
the Faculty of the Computer Science Technology Program
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
In Computer Science Technology

University of Cincinnati
College of Applied Science
March 2006
Network Protocol Development Suite

By

Brook Patten

Submitted to

the Faculty of the Computer Science Technology Program

in Partial Fulfillment of the Requirements for

the Degree of Bachelor of Science

In Computer Science Technology

©Copyright 2006 Brook Patten, All Rights Reserved

Do not reproduce or distribute this document or software in whole or in part without the express consent of the author.

____________________________________________________________________  __________
 Broak Patten            Date

____________________________________________________________________  __________
 John Nyland            Date

____________________________________________________________________  __________
 Patrick Kumpf            Date
Acknowledgement

For my fiancé, Anne, who provides my motivation and support.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgements</td>
<td>iii</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>iv</td>
</tr>
<tr>
<td>List of Figures</td>
<td>v</td>
</tr>
<tr>
<td>Abstract</td>
<td>1</td>
</tr>
<tr>
<td>1. Description and Intended Use</td>
<td>2</td>
</tr>
<tr>
<td>1.1 Statement of the Problem</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Solution</td>
<td>3</td>
</tr>
<tr>
<td>1.3 User Profiles</td>
<td>5</td>
</tr>
<tr>
<td>1.4 Design Protocols</td>
<td>7</td>
</tr>
<tr>
<td>1.4.1 XML File Database</td>
<td>7</td>
</tr>
<tr>
<td>1.4.2 Networking</td>
<td>7</td>
</tr>
<tr>
<td>1.4.3 Multimedia</td>
<td>7</td>
</tr>
<tr>
<td>1.4.4 Programming</td>
<td>8</td>
</tr>
<tr>
<td>1.6.2 Class Overview</td>
<td>11</td>
</tr>
<tr>
<td>2. Proof of Design</td>
<td>13</td>
</tr>
<tr>
<td>2.1 Parent Form</td>
<td>13</td>
</tr>
<tr>
<td>2.2 Project Explorer</td>
<td>13</td>
</tr>
<tr>
<td>2.3 Packet Builder</td>
<td>15</td>
</tr>
<tr>
<td>2.4 Protocol Builder</td>
<td>16</td>
</tr>
<tr>
<td>2.5 Stack Builder</td>
<td>15</td>
</tr>
<tr>
<td>2.6 Network Builder</td>
<td>18</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.5</td>
<td>9</td>
</tr>
<tr>
<td>Figure 1.6.1</td>
<td>10</td>
</tr>
<tr>
<td>Figure 2.1</td>
<td>13</td>
</tr>
<tr>
<td>Figure 2.2</td>
<td>14</td>
</tr>
<tr>
<td>Figure 2.3</td>
<td>15</td>
</tr>
<tr>
<td>Figure 2.4</td>
<td>16</td>
</tr>
<tr>
<td>Figure 2.5</td>
<td>17</td>
</tr>
<tr>
<td>Figure 2.6</td>
<td>18</td>
</tr>
<tr>
<td>Figure 2.7</td>
<td>19</td>
</tr>
<tr>
<td>Figure 2.8</td>
<td>20</td>
</tr>
<tr>
<td>Figure 3.1</td>
<td>21</td>
</tr>
<tr>
<td>Figure 3.3</td>
<td>23</td>
</tr>
</tbody>
</table>
Abstract

The Network Protocol Development Suite (NPDS hereafter) is a group of tools that an Information Technology professional can use to handle a variety of network protocol related tasks. These tasks include developing new network protocols, editing existing network protocols, simulating and viewing network protocols, network configuration simulation, and any combination of these tasks. The application is designed primarily as a single user application. NPDS stores all information to XML formatted files and as such it retains much functionality of a relational data model, but is more portable than a full fledged database management system such as Microsoft SQL Server.

There are three main elements to this application, the simulation engine, the user interface, and the xml file storage. The user interface can be thought of as the middle layer, since it’s function is to allow the user to direct and manipulate data from the xml file storage and feed it into the simulation engine to test data similar to the debug information a software development suite provides.
Network Protocol Development Suite

1. Description and Intended Use

The network protocol development suite (or NPDS as it will be called henceforth) is a set of tools that a user can use to design new protocols, improve existing protocols, or even build new higher level protocols off of existing lower level protocols. It can also be used as a learning tool, to simply view existing protocols in action so that a user can become more literate about the inner functioning of a protocol.

1.1 Statement of the Problem

When developing an application that requires connectivity with other computers or devices, there are many protocols that the programmer can use to make his/her applications communicate across a network. These protocols are very well established, and in fact, the most popular of them, IP, TCP and UDP have been in use for about 20 years, since the dawn of ARPANet.

Although most protocols are designed by various standards committees and universities, implementations of network protocols are developed by operating system and programming language developers. When the average programmer wants to write a piece of network software, their only choice is to rely on the protocols provided for them by their operating system or development tool. It could be that available protocol implementations don’t meet the needs of the project. It could also be said that there are many improvements that could be made on these implementations to increase efficiency, reliability, data integrity etc, but there are currently no tools available for the typical programmer to use should he/she decide to implement his/her own protocol or modify an
existing one. If a programmer wishes to develop a new protocol he/she is forced into a hybrid of software/protocol development by trial and error.

Currently there are many tools for developing network applications, but none that would enable the programmer to develop/improve network protocols, and it is this niche that the Network Protocol Development Suite fills.

1.2 Solution

The NPDS fills the gap in current development tools by providing a comprehensive interface of editors and simulators which allow a user to implement new protocols, modify existing protocols, examine existing protocols, and a number of other related tasks.

The NPDS consists of four editors, the packet editor, the protocol editor, the stack editor, and the network editor. Although these four editors provide separate functionalities, they are not isolated. They will be heavily integrated and allow the user to quickly make ties between various areas which require them. The software will allow the user to layer the protocols and assign functionalities to each layer according to the widely accepted OSI Model.

The NPDS was developed using Microsoft’s Visual Studio.Net 2003 with the 1.1 revision of the .Net CLR. This enabled it to be compiled for any .Net platform that either Microsoft or Novell Supported. In this way the software was written once, and it can be distributed to Windows, Linux, or even Macintosh Clients. Linux and Macintosh support is pending the completion of the Mono.Net CLR which is scheduled to be complete (1.2) in late 2006.
The Interface is entirely window forms driven and is similar to other development tools, in that it works as a Multiple Document Interface where the developer can have multiple editors open at once within one larger window.

The NPDS writes projects to project files for non volatile storage between sessions. Users can save, save as, and open project files much the way many other applications such as Word or Visual Studio allow.

Portions of the system which require heavy graphical output are rendered with .Net’s GDI+ graphical library.
1.3 User Profile

There are three categories of users. The categories are Application Programmers, Networking Professionals, and Students. Note that all three categories of users would be considered to be “advanced” in terms of general computer use/window navigation. They vary only by their levels of expertise in the areas of network hardware and software development.

Application Programmers

Application Programmers are users who are using NPDS to develop a network protocol for use with an application they are developing. These users are the most advanced users, since they have detailed knowledge of how software in general works. Application programmers generally use NPDS in conjunction with their application development suite such as Visual Studio or VIM/GCC. These users primarily use the suite to develop new protocols and modify existing protocols. They have working knowledge of how network hardware operates.

Networking Professionals

Networking Professionals are users who use NPDS to test different network configurations with existing network protocols. These users might have more knowledge of how network hardware works, but less knowledge about how software in general works. They might use NPDS in conjunction with a packet capture suite such as Ethereal. Primarily the Network Professional user uses the Network Designer interface.

IT Student
IT Student Users are primarily interested in viewing existing network protocols over existing network configurations. The purpose of viewing these items in action is to learn more about how they operate. IT Student users have excellent general computer skills, but might be less skilled in the areas of network hardware or software development. Typical use of the software might consist of viewing a specific network protocol over a specific hardware configuration, and watching how the virtual network operates on the data and packets.
1.4 Design Protocols

This project utilizes all four areas in information technology, those being Database, Networking, Multimedia, and Programming to differing degrees.

1.4.1 XML File Database

The NPDS stores all data in the form of xml files. Accessing the files is done using .Net’s serialization functions and as such the data model will be a relational data model much like those used with commercial DBMS systems such as SQL Server or Oracle. The difference is that more code was written to query the files, but this inconvenience is small compared to the benefit of not having to purchase, install, and configure a full fledged DBMS for every NPDS installation.

1.4.2 Networking

Obviously, the Network Protocol Development Suite makes heavy use of data structures which model a functioning network. As for connectivity to an actual network, the NPDS provides a packet capture and analyzation tool for the purpose of gathering information to build or modify the protocol within the virtual network.

1.4.3 Multimedia

The simulation tool makes heavy use of graphical renderings in order to provide the user with an understanding of what is happening on their network. This rendering is comprised of pre-rendered images and dynamically generated graphics and charts. The simulator is the most graphically intense portion of the software, and it utilizes a great deal of GDI+ to make the multimedia experience fulfilling. NPDS utilizes a consistent set of icons to symbolize each of the pieces
of the system. Each specific component has its own Icon, which appears in the project explorer, as well as the editor window which modifies it. The help system consists of “tooltips” and a help menu that is accessible in the parent form.

1.4.4 Programming

The interface has to be intuitive and user friendly for a wide variety of IT professionals such as network administrators and application developers, and as such, the interface was developed using Windows forms and Microsoft’s .Net framework. For more information regarding the programming see sections 1.5-2.
Figure 1.5 User Flow Diagram
Figure 1.6.1 Class Diagram
1.6.2 Class Overview

1. **Abstract Class Selectable** – This is an abstract class which virtually all objects in NPDS inherit from. It essentially provides two methods, Select and Deselect.

   When select is called the objects properties are gathered from the Properties Property and sent to the NPDS Project Explorer so that the user may modify them. It does this by connecting three delegates (getproperties, getpropertytypes, and setproperties) from the object to the project explorer. The Deselect method simply breaks this connection. It also provides a private static reference to the currently opened Project.

2. **Interface Renderable** – This interface is used by the Event and Device objects. It provides functionality which allows the object to be rendered to an interface and allows it to be moved around in a two dimensional plane.

3. **Interface Executable** – This interface is used by the Device, Stack, Layer, and Protocol objects. It provides methods for Starting, Stopping, Sending Data, and Receiving Data.

4. **Class Project** – This class is essentially the container for everything the user creates including Protocols, Stacks, and Networks. The primary functionality this class provides is the ability to serialize and deserialize (save and load) the object to and from a file.

5. **Class Protocol** – Contains the packet format and events which the user creates for the protocol.

6. **Class Packet** – Contains the format for data for a given protocol.

7. **Class Field** – Contains data for an individual section of data within a Packet.
8. **Abstract Class Event** – Abstract class which Classes Action, Assignment, Conditional, and Entry Point inherit from. It also provides functionality to keep track of the next event to call after the current event.

9. **Class Stack** – Contains Layers and maintains a list of all variables required to execute the current stack.

10. **Class Layer** – Contains Protocols and filters and identifying fields for each protocol.

11. **Class Network** – Contains Devices and Connections and the functionality to move data between them.

12. **Class Device** – Contains a stack and an array of network interfaces. Provides functionality to execute various stack operations and move data between the stack and it’s interfaces.

13. **Class Network Interface** – Contains a hardware address and provides the ability to send data across connections in the network.

14. **Class Connection** – Provides the ability to transport data across a network to all network interfaces it is connected to. Also provides a Maximum Transmission Unit size to a network interface when requested.

15. **Classes Action, Assignment, Entry Point, and Conditional** – All are types of Protocol Events. They each provide one input and one or more outputs. An example is the conditional, which provides one input, and two outputs. During render time both outputs are rendered. During run-time the correct output is selected based on a Condition.
2. Proof of Design

2.1 – Parent Form

This form serves two purposes.

1. To serve as a container for all windows and dialogs opened by NPDS
2. To provide a “menubar” which provides common functionalities used across the entire application such as “Save”, “Load”, “New”, “Options”, and “Run”

![Parent Form](image)

Figure 2.1

2.2 – Project Explorer

This form normally exists “docked” to the right side of the parent form, and allows the user to navigate the various aspects of their project. It also has a “properties” listview control which allows the user to edit the properties of whichever aspect of the project they have currently selected. The properties available for modification depend greatly on what object is currently selected. Note that it is NOT necessary to select an object within the project explorer to edit it’s properties. All of the editors provide the ability to select and object within their bounds and then use the project explorer pane to
edit it’s properties. To edit a property a user double clicks on the value in the properties
listview that he/she wishes to modify.

Figure 2.2
2.3 – Packet Builder

The Packet builder interface allows the user to design and modify packets for their project. From this interface a user may select a field to edit its properties, add a new field, remove a field, or move a field forward or back in the order. The “add”, “remove”, “move left” and “move right” operations are available in a context menu when the user right clicks anywhere on a field.

![Figure 2.3](image.png)

Figure 2.3
2.4 – Protocol Builder

The Protocol Builder interface provides an interface to draw working flow diagrams similar to the way MS Visio allows the user to do so. The difference is that diagrams drawn in NPDS are actually executable. When modifying a protocol a user can add a new event, remove an existing event, move an existing event, edit a selected events properties, and connect an event to other events to form the execution chain. If a user left clicks on an event he/she can move it to a different location. Right clicking and dragging allows the user to connect an event to others, and double clicking brings up a context menu which allows the “add” and “remove” functionalities.

Figure 2.4
2.5- Stack Builder

The Stack Builder interface allows the user to essentially chain protocols together and create connections between them. The user begins by either selecting an existing layer or creating a new one. Once a layer is selected the user can modify the name of the layer, add and remove protocols, and add and remove filters and identifiers for that protocol.

![Stack Editor - TCP/IP](image)

Figure 2.5
2.6 – Network Builder

The network builder provides the user with a graphical interface to build a network to simulate his/her network protocols on. By left clicking and dragging a user can move a specific device or connection to different locations (this is purely to logically separate units of the network for aesthetic value, it has no bearing on function). If a user right clicks and drags he/she can connect the selected device to other devices or hubs. Double clicking anywhere brings up a context menu. If a device is selected the context menu provides the ability to add a new interface to the device. If nothing is selected then the “add”, and “remove” options are available.

Figure 2.6
2.7 – Simulator

The Simulator form is essentially the “debugger” for the NPDS. It is really “where the rubber meets the road”. The simulator has a tab that resembles each of forms 3-6 but with additional frames which show “watches” and “locals” much the same way the MS Visual Studio does to view and modify local variables. The simulator form also contains “VCR like” controls that provide pause, forward, and stop functionalities which allow the user to control the simulation as they watch the various variables and decisions change. In each frame the user can watch as packets of data move between the various components of the virtual network.

Figure 2.7
3.1 Timeline

During Senior Design 1, the following were accomplished:

- Defined Problem
- Analyzed Requirements
- Researched Requirements
- Researched Protocol Design
- Defined Solution
- Developed proposal and oral presentation

During Senior Design 2, the following were accomplished:

- Designed Solution
- Began Implementation Phase
- Implemented Packet Editor
- Implemented Protocol Editor
• Implemented Stack Editor
• Implemented Network Editor
• Implemented Base Classes
• Implemented Console based simulator (VERY limited)

3.1.3 Senior Design 3 Accomplishments

During Senior Design 3, the following were accomplished

• Implemented Graphical Simulator
• Unit Testing
• Created IP/ICMP Implementation
• Developed final documentation and oral presentation
3.2 Software and Hardware Requirements

Software Requirements:

To Develop

Windows XP


To Use

Windows XP

Microsoft .Net Framework Redistributable

Hardware Requirements

To Develop or Use

Personal Computer (Preferably less than 2 years old, no “Hard” requirements, Newer faster hardware will operate better, but any platform should be able to execute the software at SOME speed)

3.3 Budget

<table>
<thead>
<tr>
<th>Item</th>
<th>Real World Cost</th>
<th>My Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development Workstation</td>
<td>$1500.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>Visual Studio .Net</td>
<td>$500.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>Developer Salary</td>
<td>$75,000.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>Network Hardware (switches, routers, cables etc)</td>
<td>$100.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>Total</td>
<td>$77,100.00</td>
<td>$0</td>
</tr>
</tbody>
</table>

Figure 3.3
4.1 Deliverables

1. A windows executable file (.exe) Which includes All of the Forms mentioned in the “Forms” Section.

2. Users can complete the following tasks with the software
   - Create/Edit a new network protocol
   - Create/Edit a new network packet
   - Create/Edit a virtual network
   - Create/Edit a new network stack
   - Save/Open/Close Projects
   - Simulate Virtual networks consisting of virtual network hardware running developed network protocols.

3. Example Implementation file of the IP and ICMP Protocols (IPICMP.npds)

4. Application Installer.

5.0 Testing

As with any development tool, the NPDS is capable of an absolutely enormous amount of possible outputs and configurations. Because of this, the best testing procedure is to implement protocols which are already in existence within NPDS and simulate them, then compare the output with the output of a current implementation of that protocol. My testing solution was to implement the existing IP protocol within NPDS and compare its functioning to that of an existing implementation. When these two implementations matched functionality, then it could be said that the NPDS simulation engine is stable and reliable. In regards to smaller units of functionality, unit testing was used to ensure that each module performed correctly and exhibited the expected behavior.
6.0 Conclusion

The Network Protocol Development Suite enables a user to design, develop, modify, and analyze network protocols in order to accelerate development time on networked applications, assist with network layout, and help to teach students how network protocols function. I learned a great deal in developing this software, and I will continue to develop NPDS well beyond the scope of senior design. I believe that there is commercial potential for a tool such as this, but that remains to be seen.
Appendix A: References


RFC 791, 1989, USC


www.protocols.com
http://www.protocols.com/pbook/tcpip1.htm
http://www.protocols.com/pbook/tcpip2.htm

TCP/IP Handbook
http://www.sans.org/resources/tcpip.pdf

www.networksorcery.com
http://www.networksorcery.com/enp/protocol/ip.htm#RFCs

www.cisco.com