Practical Educational Uses of the Open Distributed Network Monitor (ODNM)

**Pedrum Minaie, Jacob W. Kallman, Jason Truppi,
Sergiu M. Dascalu, Frederick C. Harris, Jr.**

**Department of Computer Science and Engineering
University of Nevada, Reno
Reno, NV, 89557 USA**

**pminaie@ieee.org, jkallman@unr.nevada.edu, jason@physics.unr.edu,
dascalus@cs.unr.edu, fredh@cs.unr.edu**

ABSTRACT

As educational institutions grow in size, network administrators for both K – 12 and higher education are finding their resources tapped for other purposes, and their staff sizes inadequate to meet the demands of ever-growing networks. The ODNM tool is presented which will allow system administrators to monitor wide area networks with both speed and convenience. In addition to this use of ODNM, teachers can also use the tool to teach their students about computer networks and network administration. Because of the basic design goals for ODNM, the tool is scalable enough to fit a wide range of higher education and school district-sized needs, as well as individual or small-scale networks. Presented in this paper are the use cases, user interface examples, and other system design elements as well as a basic introduction into the use of ODNM and future work plans.

Keywords: Network Administration, Network Monitoring, Distributed Computing, Wide-Area Networks, Education

1. INTRODUCTION

Educational institutions at all levels in the United States, from elementary education through higher education, are seeing their enrollment and class sizes grow at a fast rate. Additionally, the implementation of new technology has become a necessity to meet this growth in students. Students in classrooms of all levels now have access to an unprecedented array of technology with which to enhance curriculum. The downside of this effect is that while the class sizes have grown, and the technology implementations are growing, the staff needed to maintain this growing array of technology has not. In many educational institutions and public school districts, the ratio of computers to maintenance staff exceeds the ratio found in private companies by more than 5 times, which leads to unnecessary downtime, data insecurity, and a reduction of education quality for students [1]. It becomes clear that an efficient software solution must be found to increase the efficiency of time spent for the administrators of educational networks. To help make maintenance issues less taxing, we present the Open Distributed Network Monitor (ODNM), which uses a distributed model to help network administrators and maintenance staff get up to the minute snapshots of the status of their network. ODNM was designed with 3 basic goals in mind:

- Portability – Allow the client software to run on multiple platforms to allow administrators to monitor their networks from anywhere using their computers or PDAs.
- Scalability – Allow the server software to monitor any size network, from small one-building networks to large wide-area networks.
- Usability – Allow any type of user to get information about the network, so that assistants and maintenance staff can use the software with no troubles.

By following these three goals, the design of ODNM is one such that a small number of network technicians can keep networks under supervision even across broad geographical distances. In this way it is ideal for school district or cross-campus networks.

Additionally, ODNM can be used as an educational tool itself. As the IT industry continues to expand, a growing number of people that are new to the industry are taking networking classes to jump start their careers. Further, many students in compulsory education are also taking more network and technology classes in schools. Many of these classes introduce students to the fundamentals of networking, such as the basics of TCP/IP, network devices, and network mediums. They also teach students how to pass a specific, networking-related test such as Microsoft's MCSE, Cisco's CCNA, or CompTia's Network+[2, 3, 4]. What these classes do not usually teach students are the basics of network administration. Students many times leave classes without having the knowledge of how to use simple network administration

and troubleshooting tools or even know about the existence of such tools. ODNM can be used in the classroom to teach students the basics of network administration by allowing students to get hands-on experience monitoring networks.

These two distinct uses, one for increasing the effectiveness of maintenance staff and the other as an educational tool, are examined in this paper. Section 2 outlines the use cases and other low level design of the software package. Section 3 presents the high level design of the software, namely user interface examples and system output. Section 4 outlines the current state of development and testing. Section 5 discusses the 3 main paths of future development. Section 6 concludes the paper.

2. LOW LEVEL DESIGN

During specification and design, we used the models and elements outlined in the Unified Modeling Language (UML) [5, 6], by following the guidelines for the Unified Process (UP) [7]. UML's notations of use-cases have been used in conjunction with both functional and non-functional requirements to ensure the application's specification requirements are verified. UML has also been used during the design phase for the tool. While they are not included in the text, the functional and non-functional requirements are available upon request. Further, the use cases presented are only partial use cases, and full use case diagrams are also available.

ODNM is designed to work using a client-server model for operation. The server side is designed to run without user interaction, and is configured using a simple text configuration file, editable by the administrator, which is read at invocation. It is written using Java and Perl currently, with future development ideas moving toward a C/C++ implementation with some limited Java or Perl scripting. User interaction with the server is kept very simple, as shown in Figure 1.

As can be seen from this partial use case diagram in Figure 1, the user has no direct interaction with the server while it is running. At each time interval, which is a value set in the configuration file, the server runs a scan of the machines on the network, then stores the information in a database. The machines it scans are also specified by the administrator in the configuration file. By dividing up the work amongst multiple ODNM Server machines (machines running the server program) the administrator can run multiple scans in parallel, thus distributing the work and saving time. This allows the scans to be run much more rapidly, giving the administrators more up-to-date snapshots of network performance.

On the other hand, the client software is fully user-driven, and is the way in which an administrator can query the server while it is running. It is written solely in Java, with an eye toward interoperability among desktop operating systems. The idea was to allow administrators to access a snapshot of their network wherever they may be, even if they are not near a desktop machine. As shown in Figure 2, the administrator uses the client interface to interact with the server while it is running to get network information and display the status of the network.

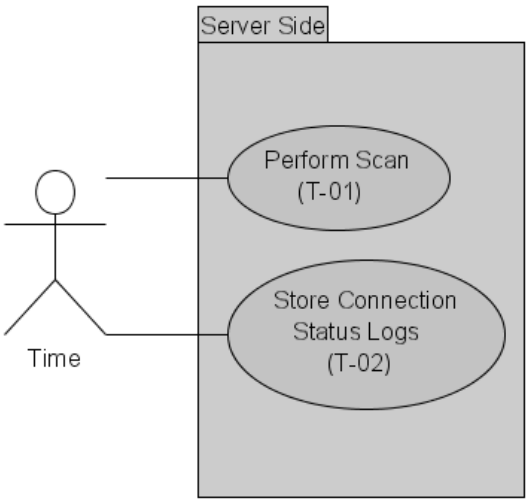

Figure 1 Server-side Use Case Diagram (partial).

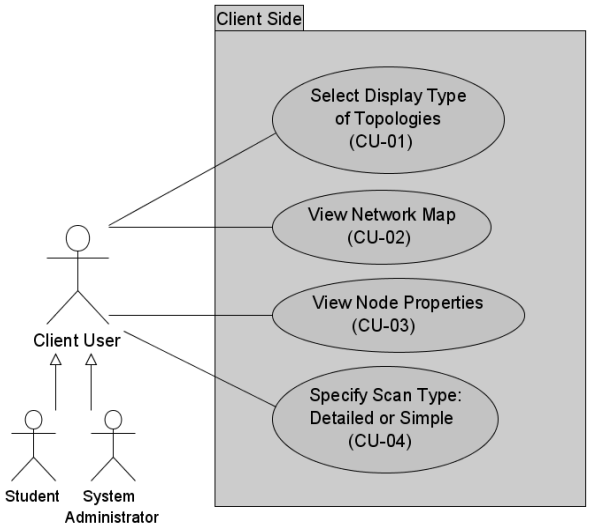

Figure 2 Client-side Use Case Diagram (partial).

Figure 2 shows that the user can query many types of information for the server while it is running, without interrupting the essential server functions such as client connection management, network scanning, and database updating, which are all run as separate threads of execution. This is how ODNM can work in a distributed

fashion, while still giving the administrator full network status information at-a-glance. The client connects to the top level server in the ODNM hierarchy, which in turn updates information from each machine it is connected to in the hierarchy. This means that the administrator only has to have one connection in the client profile, but can get the entire network snapshot from that one server. This is illustrated in Figure 3.

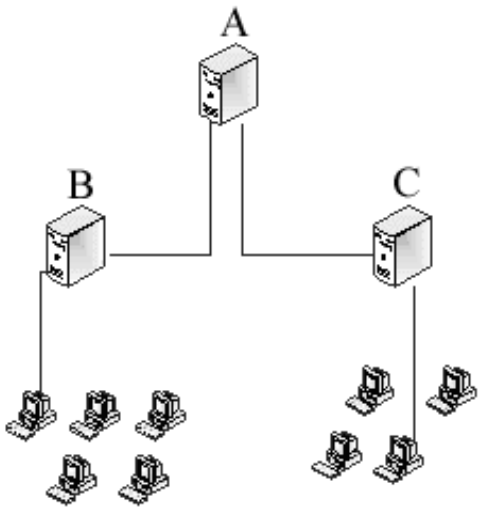

Figure 3 ODNM Server Hierarchy

Figure 3 illustrates how an administrator could set up a server hierarchy. Machine A would be the root server, and would therefore maintain the connection to the outside world with which the client server would connect. It would scan directly Machines B and C, but not the machines below them. Machine B would itself scan the machines below it, and Machine C would scan the computers below it. Machine A would then take the three connection information databases from itself and Machines A and B, and integrate them into one database that could be queried by the client software. In this way the scans are done in parallel, and the database updated on Machine A. This saves time and allows the client to get information from one source, instead of having to connect to all servers in the hierarchy.

By using the distributed model the network can be scanned in a very rapid fashion, while also adding a high level of scalability to the software. As the size of the network grows, more servers can be added to the hierarchy to increase the distribution of tasks. This will give administrators flexibility in their scanning routines. Further, by allowing one point of connection for the server, we allow the portability of having one server platform, but having a customizable client interface. This will be discussed further in Section 5, where we will discuss the porting of the client software to many different platforms including web-based interfaces using J2EE, Palm/PDA platforms, and other operating systems and environments. These features will make the tool very usable, and will help ease the strain on administrators.

Furthermore, students can learn about many aspects of applied computing by experimenting with the tool. First, by configuring the server, students would get experience editing scripting files and configuration files, which is an essential skill for network administrators. Second, by structuring the server hierarchy, students would get exposure to network topologies, and distributed computing. Thirdly, using the client software would get students experience working with Java applications, and also connection management. Finally, the entire experience of using the tool would give them a broad overview of the importance of system monitoring, without having to grind through pages of generated log files.

3. HIGH LEVEL DESIGN

ODNM is designed to minimize information presented to the user, so that they may concentrate on pertinent information and get that information quickly. For this reason a graphical client interface was essential. On the server side, however, the output is confined to a series of text files, which can be read by the administrator if desired.

Figure 4 gives an example of some of the information that the server scans for at each interval or on-demand scan. The IP address of the machine scanned, hops between server and scanned machine, speed of connection, open ports, operating system, and other information gathered during scanning are displayed, as well as some flags such as SLOW_NODE, NEW_NODE, and other types of warnings are stored. The amount of information passed to the client is configured by the administrator from the client interface.

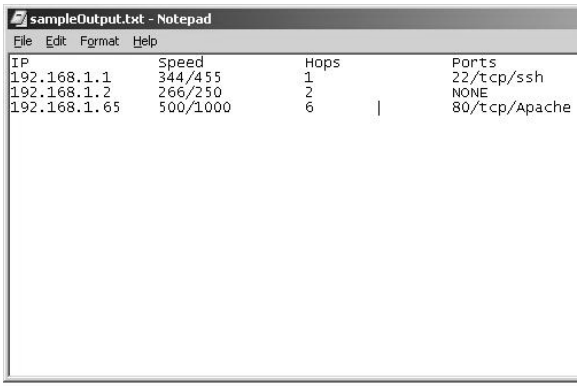

Figure 4 ODNM Server Scan Database File (partial).

The server code itself focuses more on the overall management of the tasks, instead of the scanning. It uses the Nmap program to do all of the scanning [8]. We

implemented the server in this way because it allowed us to focus on the management of multi-threaded tasks, as well as the overall implementation. As discussed in Section 5, we hope to implement our own scanning routine in the future, and the code has been modularized to allow for the Namp scan module to be replaced by our own code.

The client interface itself is designed to give the administrator full status information about the network without cluttering the screen and impeding readability. As shown in Figure 5, the interface gives a quick snapshot of the network status, with side fields showing more detailed information.

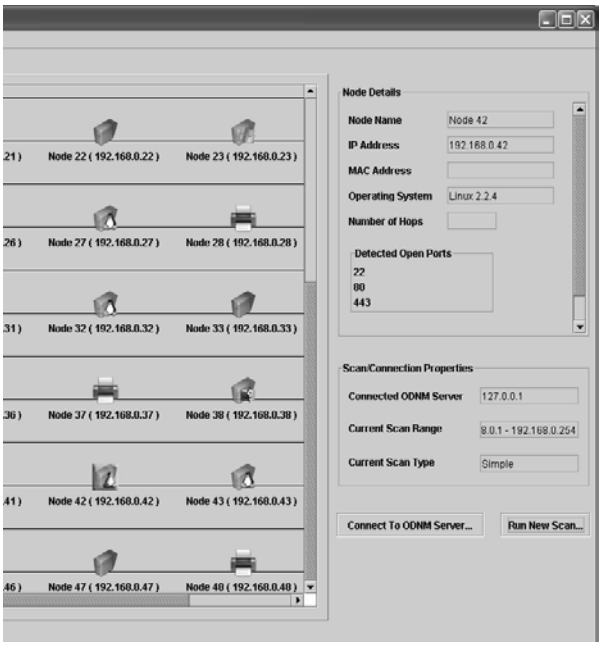

Figure 5 ODNM Client Interface Window (partial).

To the left of the screen there is an icon to represent the different machines on the network. To the right there is a text box which will display the information for the node that is clicked on in the left box. The bottom right shows the last type of scan, and has buttons allowing the user to initiate new scans or reconnect to servers in the hierarchy. In this way, the interface can display a lot of information without overloading the user or cluttering the view. Also available as tabs on the left hand box are tree view, which eliminates the icons and just shows a tree structure for the nodes, or physical view, which will allow an administrator to view a GPS-enabled physical map of the network. This is be useful in determining geographical points of failure in wide-area networks. The GPS feature is not implemented as yet, and will be discussed in Section 5.

Overall, ODNM is structured to balance simplicity with usability. The server side is designed to run as efficient as possible, without the overhead of a graphical interface. The client, on the other hand, sacrifices some speed for usability by giving the user a full-featured graphical interface. This will give administrators the ability to view network information at-a-glance, while not overwhelming non-administrators, or students who are using the tool from a beginner level. This would also allow students to experience two different types of program environments, the graphical user interface environment, and the command line user interface environment.

4. CURRENT STATE OF DEVELOPMENT AND TESTING

ODNM is currently fully functional to our initial requirements. We have tested it on various sized networks ranging from 5 addresses to 254 addresses (10 machines and 244 empty addresses). These tests were informal and we have not entered a formal testing phase. Currently, we are focused on cleaning up and fully commenting the code, as well as writing user documentation and instruction files. Once we have what we feel to be stable, secure, and complete code with good user documentation and troubleshooting we will enter the formal testing period.

The tool itself is not ready to be used in an actual live environment, and much of the work in the coming months will be focused on authentication protocols and security. Since the server listens on a port that must be opened to the outside world so that the client may connect, authentication of users and connection security must be implemented before the system can enter a real environment. Some of these problems could be simplified by going to a web-based client using the J2EE platform for web applications [9], however we are planning to implement authentication and security features into the current client model.

In the short term, we will also be organizing a more convenient way to distribute and update new versions of code. We will be moving to a new website for the project, on our own domain, and also be organizing maintenance and development channels within the ODNM team. This will allow us to update and maintain code more efficiently, while also giving us a better path to versioning and program releases.

5. FUTURE WORK

The future work on ODNM is organized into 3 main subclasses: Essential Updates, Algorithmic Updates, and Feature Updates. These reflect the main areas of application improvements for ODNM.

Essential Updates
This category of future development work reflects essential program features that must be included before the next version of ODNM is to be considered complete. It is perhaps the biggest category of future work, and included structural and cosmetic upgrades and improvements. A partial list of the goals under this category is as follows:

- Refine the database updating routines for the ODNM server software. Efficiency and debugging are the main concerns.
- Implement necessary security and authentication in the server connection management routines. This includes user authentication and secure input buffering.
- Complete the user documentation, both in-code and general information.
- Implement a web interface using J2EE technology. Implement a simplified web interface to the server. This may or may not be a highly-utilized function, but would be useful in situations where client functionality is not available.

These are just a sampling, and most of the future work in this class would be considered basic testing and debugging to ensure stability and efficiency under all operating conditions.

Algorithmic Updates
This category of future development deals with new functional development for ODNM, and is mostly centered around improving already existing modules of the code. It is focused on two main areas: True Distribution and Scanning.

True Distribution: Currently, ODNM relies on the system administrator to configure which server scans which block of IP addresses. In this way it is a user-defined distribution of tasks. In the future we would like to implement a more intelligent form of workload distribution. We would like to have the root node work as a workload manager and delegate out the scanning to machines based on true distributed computing algorithms. This way the system administrator would not have to do as much work up front, and the ODNM Network would behave more intelligently. This would most likely also increase efficiency and reduce scan times, as the differences in hardware and network resources would be dealt with.

Scanning: Currently, ODNM relies on Nmap and ping to get information about the network. The problems with this are twofold. First, it increases interdependence on other applications and restricts interoperability through the need for system calls and output parsing. This means, in short, that the system won't work in Windows environments without substantial code rewriting due to differences in directory structures and program output. Second, the network scans are very intrusive, and use active pinging and scanning to get information about the network. This leads to more network resource usage, and a reduction in both speed and efficiency. We would like to aim for a new scanning routine which could get the information we wanted while remaining platform independent, and also be more passive than it is currently. We feel that if we write our own routines, instead of system calling to current routines, we can gain passiveness and efficiency in our software, while remaining platform independent on both the client and server sides. This would lead to faster execution of scans, more real-time information gathering ability, more portability, and a better product.

These updates would require a sufficient amount of time be invested, both in research and in testing. Because of this, we are pushing them off until the Essential Updates are complete and we have a product that is ready to be tested as is. Additionally, we will be looking for additional people to develop parts of this updates who have more experience in the realms of distributed computing and network scanning.

Feature Updates
This category of updates includes features which are not essential to the software package, but would make the package more useful. Some of these are as follows:

- GPS-enabled physical mapping, as discussed before. The groundwork for this feature is already implemented, both in the server configuration and client interface, the mechanics just need to be implemented.
- Automatic software updating ability. This would allow network administrators to update security patches, system upgrades, etc., without having to visit each machine on a network. This is already a feature of many network administration tools, and we would like to integrate it into ODNM to further ease the strain on administrators.
- Scan history logging. Implement a more extensive log file and scan history capability with e-mail, instant messaging, and file alerts. While this is not essential on many systems, which keep their own logs, it would again integrate more functionality and give administrators simplified access to information.
- Client-side portability. Implement versions of the client software to run on PDAs, and other types of mobile computing solutions to give administrators more flexibility and mobility in monitoring wide-area networks like school districts.
- Server-side portability. Scale down the server and increase efficiency so that it can be run on

embedded devices like managed switches and routers. This would allow ODNM to give more accurate network topology mapping abilities.

This list is not all inclusive, but is intended to show the inclusion of more features is consistent with the plan of making ODNM a portal through which to access a large amount of network information within the framework of an easy-to-use, readable interface.

Overall, our goal for ODNM is to allow system administrators to streamline their time spent maintaining the network. By giving them one-touch access to system upgrades, administration, and security alerts, we hope that ODNM will reduce downtime and data corruption. This would lead to a direct increase not only in administrator satisfaction, but also the quality of student education.

6. CONCLUSION

As illustrated, ODNM will give system administrators the power to maintain and upgrades networks on all scales with ease and thereby reduce the impact of budget and manpower deficits. However, since ODNM is built to be easy-to-use, it will allow beginners to learn about network administration and many aspects of applied computing. It will allow students to have hands-on experience in workload distribution, network security monitoring, and other topics.

ODNM's main advantage to the students over most network scanners is its ability to graphically view all the devices within the entire network. ODNM's graphical network map will allow the students to gain a visual perspective on the scanned network whereas the students can be overwhelmed, perhaps even confused, by the output of a text-based network scanner. Using ODNM in the classroom can further students' knowledge of networking when used to accompany assignments that mimic real-world scenarios. Certainly, curriculum at all levels can be developed with which ODNM can assist. As an example, one assignment may be to locate open ports on specific computers in a network, assess their security threats, and discuss options that are available to reduce those security threats. Or, the students may be presented with a scenario in which a virus is propagating quickly throughout a multi-subnet network and its distribution must be stopped immediately. Furthermore, the students will have to conduct research on the vulnerability, use network administration tools to locate vulnerable systems, and take appropriate actions. From these assignments, the students will be able to gather information for a given network, investigate the network devices in question, assess their security vulnerability, and would be able to convey his or her findings to the proper people within an organization with the aid of ODNM. These types of assignments can be expanded into other topics that we have discussed as well.

ODNM, therefore, would be an asset to any educational institution, public or private. It would also be useful to individuals interested in learning more about network security, or interested in monitoring their private network.

7. REFERENCES

1. Washoe County School District Strategic Technology Plan, Section 8. July 2003.
2. Microsoft MCP 2004). Available as of April 22, 2004 at: http://www.microsoft.com/mcp
3. Cisco CCNA (2004). Available as of April 22, 2004 at: http://www.cisco.com/en/US/learning/le3/le2/le0/le9/learning_certification_type_home.html
4. CompTIA Network+ (2004). Available as of April 22, 2004 at: http://www.comptia.org/certification/network/default.asp
5. Jacobson, J, Booch, G., Rumbaugh, J.: The Unified Software Development Process. Addison-Wesley (1999).
6. OMG's UML Resource Page (2004). Available as of April 21, 2004 at: http://www.omg.org/uml
7. Booch, G., Rumbaugh, J., Jacobson, I.: The Unified Modeling Language: User Guide. Addison-Wesley (1998).
8. Nmap Security Scanner version 3.50 (2004). Available as of April 21, 2004 at: http://www.insecure.org
9. Java 2 Platform, Enterprise Edition (J2EE). Available as of April 22, 2004 at: http://www.java.sun.com/j2ee/index.jsp.
10. HP OpenView Network Node Manager 6.4 and Network Node Manager Extended Topology 2.0 (2004). Available as of April 22, 2004 at: http://www.openview.hp.com/
11. NetRadar (2004). Available as of April 22, 2004 at: http://netradar.sourceforge.net/
12. LANSurveyor 8.0 (2004). Available as of April 22, 2004 at: http://www.neon.com
13. Subramanyan, R., Miguel-Alonso, J., Fortes, J.A.: A Scalable SNMP-based Distributed Monitoring System for Heterogeneous Network Computing. Proceedings of the ACM/IEEE Conference on Supercomputing, Dallas, Texas, USA (2000).
14. Nmap (2002): Remote OS Detection via TCP/IP Stack Fingerprinting. Available as of April 21, 2004 at http://www.insecure.org