MiNet: Managing Internet Connectivity in Academic Computing Labs

Shaun M. Lynch, Ph.D.
Department of Mathematics and Computer Science
University of Wisconsin-Superior
Superior, WI 54880
slynch@uwsuper.edu

Abstract

Educators who teach coursework that rely on computer technology in labs designated for instruction often have to compete with the novelty of the internet for student attention. As a faculty member and administrator of the computing infrastructure in the Department of Mathematics and Computer Science at the University of Wisconsin-Superior, one of the most common requests made by instructors is how to disable internet access to lab computers during scheduled class times. The author had an insight how to resolve this problem and built a networking appliance called MiNet using open source software and technology that overcame many of the challenges associated with managing network traffic in an academic computing lab. This paper discusses the design and technologies used to resolve this issue along with ways to deploy the application in academic departments seeking a relatively elegant way to manage a difficult classroom problem.

1 Introduction

Educators who teach coursework that rely on computer technology in labs designated for instruction often have to compete with the novelty of the internet for student attention. News, email, sports results, social media are accessible in an instant and provide immediate gratification to the distracted mind. It is convenient to think that students should have the discipline to refrain from these activities during class and pay attention, but the reality is that students are still learning self-discipline and suffer the same challenges we all face at one time or another.

As a faculty member and administrator of the computing infrastructure in the Department of Mathematics and Computer Science at the University of Wisconsin-Superior, one of the most common requests made by other faculty and academic staff is how to disable internet access to lab computers during scheduled class times. Ideally, instructors would like an on-off switch that deactivates or at least limits internet access without totally disabling the local network access they rely upon. If a solution were only that simple, every computer lab used for instruction would have a switch installed!

During Fall Semester 2013, the author had an insight how to resolve this problem and built a networking appliance called MiNet that relied solely on existing open source software and technology. The solution overcame many of the technical challenges associated with managing network traffic and can be easily replicated. In addition, the application allows individual instructors to control internet access in broad strokes or at fine-grained detail without disrupting other labs or users. The approach requires modest hardware capabilities and can be easily virtualized and integrated into an Active Directory environment and managed using group policy if desired.

This paper puts forward the design and technologies used to solve this problem. Discussion begins with a background that provides an overview of mechanism used to regulate network traffic followed by a simple model used to illustrate the difficulties of managing and controlling network traffic in an academic computing environment. Next, the MiNet application is presented along with details that cover configuration, core functionality, deployment, and browser support necessary to manage internet access in a computer lab. The paper concludes with ideas how to improve the application and to simplify deployment so that it can be adopted by other academic departments seeking a relatively elegant way to resolve a difficult classroom problem.

2 Background

Managing internet access in classrooms and labs populated with computer workstations requires the coordination of interacting technologies to be effective. The problem yields a number of challenges that include providing independent control across multiple labs, scheduling between classes, access and security, accounting for human error, browser support, as well as, a number of other issues all of which incur tradeoffs that must be considered.

It is important to acknowledge that preventing students from surfing the internet during class time on university owned lab computers is only one step toward addressing the plethora of distractions students engage in. Student owned technology like smartphones and laptops contain wireless technologies capable of connecting to networks beyond the control of the local network administrator. Regardless, computing labs are a shared resource and need to be managed effectively to create an environment conducive to learning.

2.1 Regulating Network Traffic

A firewall refers to a class of devices that control incoming and outgoing network traffic based on a set of rules. A firewall can be a discreet appliance or an application embedded on a host system. Firewalls generally work by monitoring traffic through one or more network interfaces and track settings in the communication protocol header.

A proxy server is a type of firewall that serves as an intermediary between one or more clients and one or more servers. Also known as an application firewall or web proxy, a proxy performs on behalf of a client when attempting to communicate with a server. When a client requests content from a server, the proxy intercepts and evaluates the request against a set of rules. If valid, the proxy issues a new request to the server in place of the client. Upon receiving a reply from the server, the proxy evaluates the response against a set of rules. If valid, the proxy delivers the server response to the client.

The rules that govern proxy behavior generally apply at the application level of the Open Systems Interconnect (OSI) model. This layer is most closely aligned with programs users interact with on a day-to-day basis that use protocols like HTTP, HTTPS, and FTP to transfer data over the network. Proxy servers may also incorporate rules that track lower layer protocols that include IP, TCP, and UDP. In the world of practice, proxy servers often work in conjunction with other types of firewalls that provide specific functionality (e.g. stateless and stateful firewalls) to enhance their overall security.

Proxy servers often contain additional capabilities important to network management that include content caching, content filtering, client anonymity, and reporting. Content caching improves network performance by storing frequently accessed content locally without having to repeatedly contact the server. Content filtering ensures data sent over the network complies with acceptable use policies. Client anonymity hides the identity of client devices residing on internal networks from external observers. And reporting organizes and collates proxy activity to facilitate administrative oversight.

These characteristic make proxy servers an ideal candidate to manage and control network traffic to a pool of computing resources since it is transparent to the user under normal operating conditions. Only in circumstances where the network communication fails to comply with policy does the device makes its presence known. In that case, the proxy server may simply drop the communication or provide a message to the user indicating the rationale for denying the communication.

2.2 Illustrating the Problem

To fully illustrate the challenge of configuring a system to manage network traffic for an academic computing lab, it is useful to look at a series of network configurations that exemplify the problem at hand. Consider the scenario illustrated in Figure 1 where a proxy serves as an intermediary between a pool of intranet residing computing resources and servers on the internet. If one assumes consistent and homogenous traffic patterns over time—a reasonable assumption in most cases, the proxy would require only a single configuration deployed by the network administrator to manage network traffic.

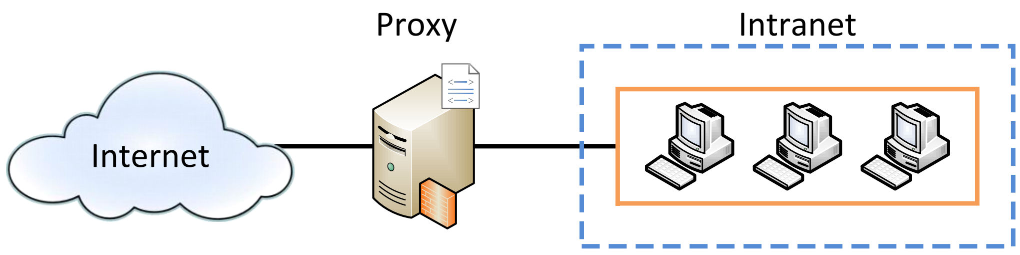

Figure 1: Scenario using a single proxy server to control and manage a single pool of computing resources.

This scenario can be extended by adding additional pools of computing resources as shown in Figure 2. Each pool may represent different functional units in an enterprise or computer lab in an academic environment. As the number of pools increase, so will the physical resources—processing, networking, memory, and storage—to host the proxy server. If the new pools being added have similar traffic patterns, it may be possible to create a single proxy configuration that satisfies all pools. If however, each pool has a distinct traffic pattern, then creating a single, monolithic configuration that encompasses the nuances of each pool becomes increasingly difficult.

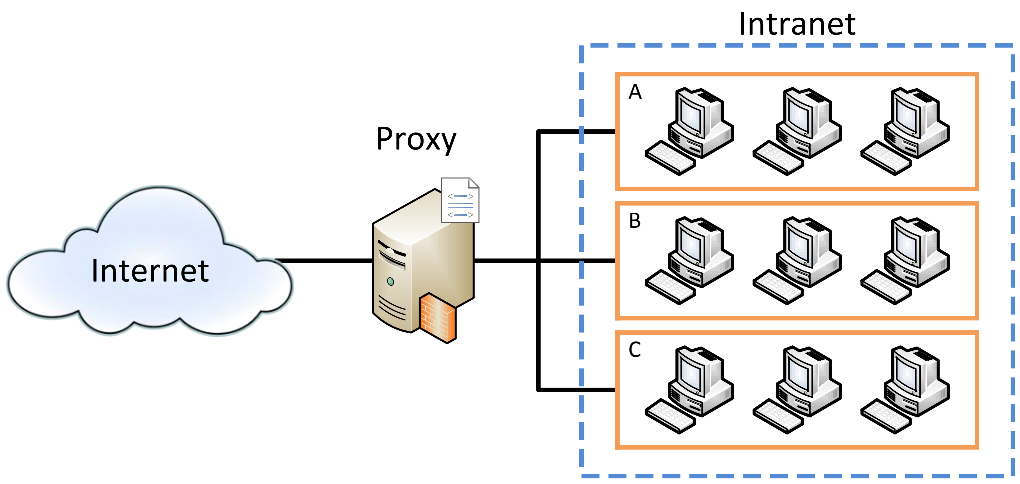

Figure 2: Scenario using a single proxy server to control and manage multiple pools that share similar network traffic patterns.

An alternate solution may entail assigning separate proxy servers to each new pool as shown in Figure 3. In this case, individual proxy servers can be configured to satisfy the specific physical characteristics and traffic patterns associated with each pool. In the past, each proxy would have been hosted on a separate physical server thus detracting from the overall benefit of such a setup. Through virtualization, however, proxy servers can be hosted on one physical host (server or cluster) and resources shared across virtual machines or containers. Additional benefits include: simpler configurations, scalability, traffic isolation, and failure compartmentalization.

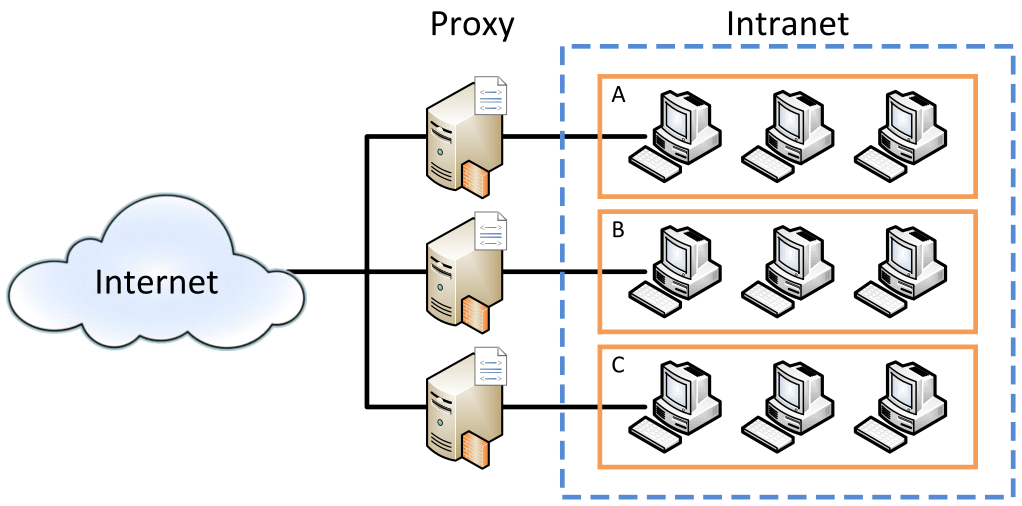

Figure 3: Scenario using multiple proxy servers to control and manage multiple pools with dissimilar network traffic patterns.

To complicate matters further, consider the scenario where there are multiple pools of computing resources whose traffic patterns within each pool changes over time as shown in Figure 4. This scenario most closely represents computing labs in an academic environment where activities constantly change throughout the academic year. Changes in network traffic patterns may stem from differences in course content (e.g. introductory computer applications versus object-oriented programming), variations in class activities (e.g. testing versus lecture), or changes in class schedules.

It is this scenario that makes controlling and managing internet access to academic computing labs so difficult. Proxy applications often use static configurations to define the rules used to control and manage network traffic. In normal circumstances, proxy configurations are prepared by network administrators and maintained periodically over the life of the server. In this situation, a dynamic proxy configuration is needed that allows authorized users to change settings base on how the computing resource is being used.

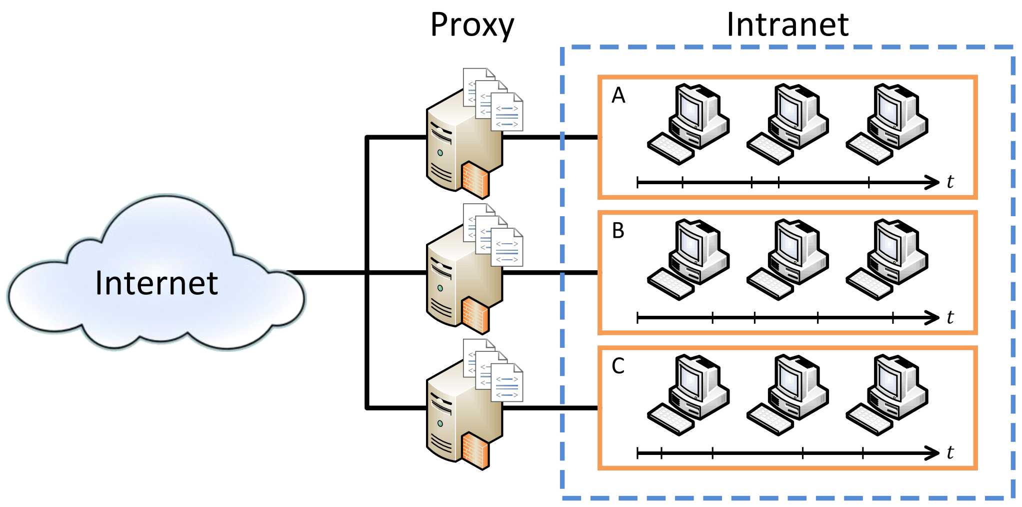

Figure 4: Scenario using multiple proxy servers to control and manage multiple pools with network traffic patterns that change over time.

3 MiNet

The author designed MiNet to enable authorized users to dynamically configure a proxy server to meet the needs of an academic environment. The application consists of a suite of scripts built upon a Squid proxy server. As an introduction, Squid is an open-source proxy application used to filter and cache web content [1]. The application is normally installed on Linux-based operating systems and uses a static configuration file to define the rules used for filtering and caching. MiNet manages the state of Squid's configuration file by responding to events that may stem from user input and schedule triggers.

3.1 Configuration

MiNet partitions proxy configuration settings into the file hierarchy shown in Figure 5. The root of the hierarchy begins with the Squid configuration file located in Squid's configuration directory. Using the *include* directive, settings are expanded into separate files that contain default rules that can be modified to provide the desired functionality.

To begin with, rules declared in the *computers.acl* file identify client workstations within the scope of the proxy server. By default, systems matching the IP addresses or address ranges listed in *computers* file are included within the proxy's scope. In addition, client workstations must have web proxy settings assigned to the respective proxy server.

Next, rules declared in the *websites.acl* file allow essential websites when the proxy is in restricted mode. By default, websites are identified by domains listed in the *websites* file. Web domains for teaching and class-related resources such as learning management systems are generally included.

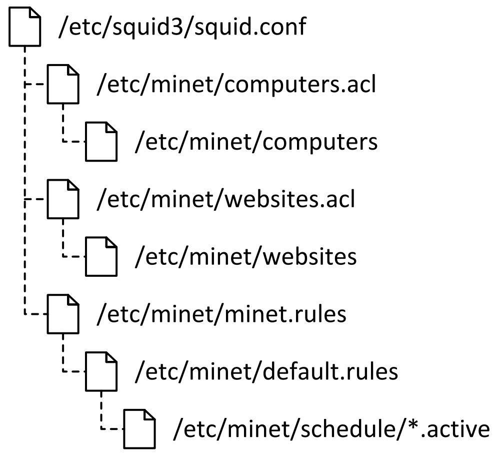

Figure 5: MiNet configuration file hierarchy.

Internet access is determined by rules defined in the *minet.rules* file. This file is automatically updated to reflect the state of the proxy server and should not be modified by hand. However, default internet access rules are declared in the *default.rules* file and can be customized to meet particular needs. The *schedule* subdirectory contains proxy rules created using the *new-scheduleitem* command. The command requires a unique name that identifies the schedule item along with optional starting and ending dates and is generally used to add default internet access rules for classes throughout the semester or academic year.

3.2 Core Functionality

The primary function of MiNet is to manage the dynamic configuration of the proxy server as depicted in the state diagram shown in Figure 6. Booting the host server will automatically start the Squid service which enters the *Use Default Access Level* state. Authorized users can change the state of the proxy by logging into the server using PuTTY and enter commands directly on the command line.

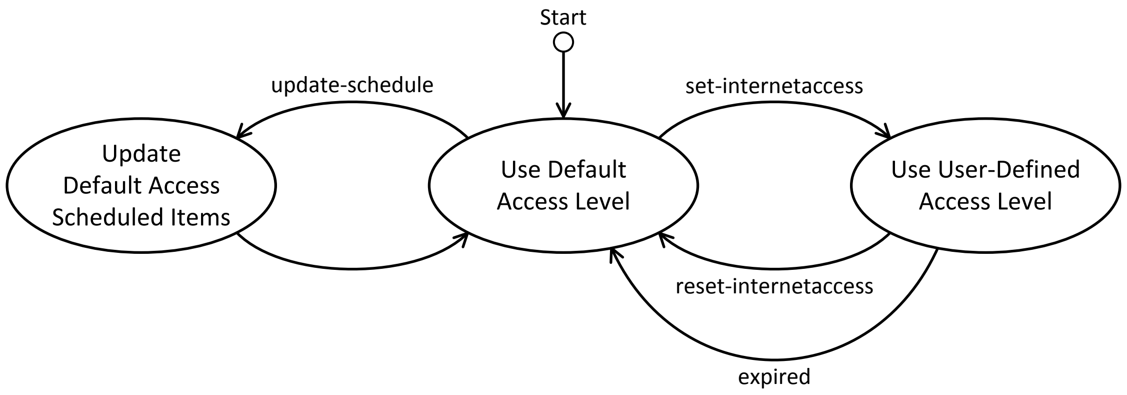

Figure 6: MiNet state diagram.

Under normal operation, the *Use User-Defined Access Level* state can be entered using the *set-internetaccess* command. Arguments for this command include *enabled*, *disabled*, and *restricted*. Restricted is a hybrid mode that only allows access to web domains listed in the *websites* file discussed previously. An optional expiration can also be included in the form of a Linux *at* expression. If the expression is invalid, an error is posted and the command terminates.

In the case where no expiration is provided, a default interval of 30 minutes is used. This prevents the defined access level from being enforced indefinitely if one forgets to reset the access level. It also provides a set-and-forget option for users who want immediate control over internet access with the convenience of a quick setting. The default interval was selected so that expected expiration would occur within one class period to prevent disruption in a following class.

In the case where expiration evaluates to an interval greater than 120 minutes, the command posts a warning and terminates without setting the access level. This prevents the situation where an access level is set for an abnormally long period. This is particularly important if the lab is left unattended after setting an access level that extends beyond normal class times. There may be instances, however, where the expiration must exceed the maximum. In this case, an override switch is provided that can be set to force an expiration interval beyond the maximum.

Authorized users can update or reset internet access at any time from the command line. Entering the *set-internetaccess* command will simply replace the old setting if another user-defined access level is in effect. Alternatively, entering the *reset-internetaccess* command will exit the user-defined access level and return the proxy to its default access level.

Entering the *Update Default Access Scheduled Items* state is primarily controlled by an entry place in the /etc/*cron.d* directory that invokes the *update-schedule* command. The command is scheduled to run at midnight every day and flags scheduled items as active or expired. The state can also be entering by adding new schedule items or by manually invoking the command.

3.3 Deployment

Proxy server access and security is paramount since a proxy server is an internet facing device and a potential target for internal and external attacks. Installing a software firewall on the device is an essential first step. Fortunately, only two network ports are required for use and include Port 3128 for the Squid proxy service and Port 22 (SSH) for remote access. Ports should be only accessible from the devices on intranet. Changing the default port for the Squid proxy service is also recommended since it is a widely recognized signature that a proxy service is running.

Account management is another area that requires special consideration. In this particular implementation, authorized users enter commands directly from the command line to change internet access settings. The simplest way to provide access to authorized users is to provide local accounts and place those users with access into a separate group. Unfortunately, this can entail a significant amount of duplicate effort if multiple proxy servers are being hosted. If the proxy server resides in a Windows managed environment, then authenticating against Active Directory greatly simplifies account management.

In this particular implementation, the *set-internetaccess* and *reset-internetaccess* can be called by any user. However, each command checks for root access before calling the MiNet system level scripts *set-internetaccess.sh* and *reset-internetaccess.sh* used to change and reload Squid configuration files. Adding an entry in the *sudoers* configuration file allowing selected users or groups root privileges to run these scripts effectively provides access to only those authorized to use the service.

The author paid special attention to embed messages that would be clearly visible to the user when interacting with the application. Using the message-of-the-day mechanism provides a simple way to display a splash screen when the user logs in containing helpful command reminders. In addition, all commands contain help messages accessible using the commonly recognized *-h* or *--help* command switches.

3.4 Browser Support

Workstations in the pool managed by the proxy must be configured to send internet traffic through the proxy server to function properly. Determining which browsers to support is first decision an administrator must make since it impacts the manner in which settings are deployed and secured. Fortunately, the most popular browsers (Internet Explorer, Mozilla Firefox, and Google Chrome at the time of this article) use the same proxy settings in a Windows environment. However, each requires special configuration to protect settings from being viewed or changed.

Configuring Internet Explorer on systems running Windows is a good place to start since the configuration can be deployed using group policy. On the other hand, Firefox uses local configuration files that must be deployed to individual workstations. While Google Chrome uses both a local configuration file and custom group policy settings that can be added to the domain controller. Regardless of which browsers are supported, it is important to test each configuration to ensure web traffic is indeed going through the proxy server.

Once the supported browsers are deployed it is critical to monitor systems for rogue installations that allow a user to bypass proxy settings. This creates a security breach and potentially offers savvy users a way to circumvent network controls. Students that do this gain an unfair advantage in classes that restrict web access to prevent unauthorized web access during tests. Google Chrome can be particularly challenging in this regard since it uses its own installer and can be installed even if Windows Installer is disabled and application installations require elevated privileges. The only sure way to prevent Google Chrome from installing is deploy a group policy that explicitly prevents the Google Chrome installer and ancillary applications from executing.

4 Parting Thoughts

Disabling internet access to an academic computing lab presents itself as a deceptively simple concept but belies the true nature of the problem and the complexity it creates for network administrators in academic computing environments. However, finding solutions to these “simple” problems is one of many challenges individuals in technology-related disciplines thrive upon and highlights the interconnectedness of all the system involved.

MiNet offers a solution for the challenge of managing internet access but its implementation is far from complete and there is room for improvement. It is author’s opinion that the first area to improve upon is the method to access and configure settings. Although there are only two commands to manage internet access, users have to login into the server and type the command on the command line. A small control panel on the desktop or a live tile displaying the state of the proxy would be more informative and much simpler to use.

Containerizing the application using Docker or similar application is another area to consider and has the potential to significantly enhance deployment. In many regards, a MiNet enabled proxy server is well suited for hosting in a container since there few, if any, external dependencies or fixed constraints. Unlike hosting each instance on separate virtual machine, containers offer a very efficient way to share common resources across multiple instances. The biggest concern is ensuring true isolation across containers to prevent the spread of a breach if it were to occur.

In summary, this paper presented a design and implementation of a novel application that enable authorized users to dynamically configure a proxy server to meet the needs of an academic environment. Topics discussed included the need for the solution, the nature of the problem along with limits of current technology, and an overview of the mechanics necessary to build the application. Using open source software and technologies, the solution provides a number of capabilities that facilitate control of internet resources and management of academic computing labs while promoting the effective use of computing technology.

References

[1] Squid: Optimising Web Delivery (2015). squid-cache.org, http://www.squid-cache.org.