2011

Netflow-Based Bandwidth Auditor for the RESNET Network

Jessalyn Schweitzer

University of Northern Iowa

NETFLOW-BASED BANDWIDTH AUDITOR FOR THE RESNET NETWORK

A Project

Submitted

in Partial Fulfillment

of the Requirements for the Designation

University Honors

Jessalyn Schweitzer

University of Northern Iowa

May 2011

RESNET BANDWIDTH AUDITOR

This Study by: Jessalyn Schweitzer

Entitled: NetFlow-Based Bandwidth Auditor for the ResNet Network

has been approved as meeting the thesis or project requirement for the Designation

University Honors.

4/29/11	[illegible]
Date	Dr. Ben Schafer, Honors Thesis Advisor, Computer Science
5/6/11	[illegible]
Date	Jessica Moon, Director, University Honors Program

Acknowledgements

I would like to thank my advisor, Dr. Ben Schafer, for guiding me along the thesis process and providing valuable feedback and advice. I would also like to thank ResNet and ITS-Network Services for providing access to the resources necessary for completing this project.

Introduction

Bandwidth consumption has become an important issue in recent years, especially on college campuses. Bandwidth is a measure of the amount of data that can be transmitted over the network within a certain period of time (Peterson & Davie, 2007). Students spend a considerable amount of time on the Internet for a growing number of purposes. Facebook, Netflix online streaming, filesharing, and other popular technologies consume not only students' time, but bandwidth as well. Network administrators need a way to monitor bandwidth consumption to ensure bandwidth is being fairly distributed among all users and that availability is meeting demand.

ResNet is the Internet access provider for students living on-campus at the University of Northern Iowa (UNI). The purpose of my thesis project was to develop a system that would allow ResNet and Information Technology Services (ITS)-Network Services to audit bandwidth usage on the ResNet network in order to enforce bandwidth usage policies. The scope of my project included configuration of software that collects and stores bandwidth data, and development of a web application to display the bandwidth usage in a web interface. This project integrates with the ResNet registration system to associate usage data with a particular user registered on ResNet. In addition to enforcement, this project also serves an educational purpose by allowing ResNet users to view how much individual bandwidth they have consumed.

Various bandwidth monitoring software solutions are in existence today. However, these products report bandwidth usage based on IP address. This is not beneficial for the ResNet network because registration for access on ResNet is based on MAC address. They also do not provide a way to associate a device with a specific user or aggregate usage for multiple devices registered to a user.

I chose this project because of the benefit it would provide for ResNet and ITS-Network Services. As an employee of ITS-Network Services and ResNet for the past two and four years, respectively, I care about students being satisfied with the Internet service we provide. The amount of bandwidth consumed by the top bandwidth users on ResNet is significant. This project has the potential to help improve network performance and customer satisfaction by enforcing fair bandwidth usage. I was also drawn to the web application aspect of this project. I will be developing web applications in my career after graduation, and this project seemed like a great way to develop those skills.

Methodology

NetFlow Data Collection

The first phase of my project was data collection. The underlying technology I used to collect bandwidth data is called NetFlow. NetFlow is a network protocol used to collect data about flows sent over the network (Enterasys Networks, 2010). A flow is a sequence of data packets containing the same source or destination address (Peterson & Davie, 2007). In simpler terms, it is essentially a conversation between two devices. In order to use NetFlow, it must be enabled and configured on a network switch or router to export flows to a collector server. The collector server uses some form of software to listen for flows and process them as they are received.

The first approach I considered was to implement my own collection software. After researching how to receive and process the flows, I decided this was beyond the scope of my project. There are several NetFlow collectors already in existence. Some require a commercial license, whereas others are free to download and install. Many of the free collectors had poor documentation and seemed difficult to maintain. The collector I chose to use is Plixer Scrutinizer. Scrutinizer is a tool used to collect NetFlow data and display a variety of graphs and reports in a customizable web interface. Though this product requires a commercial license, it will provide benefit outside of the bandwidth auditing done for my project. A vast amount of information can be derived from NetFlow data, which will help network administrators monitor and analyze network traffic.

The key component Scrutinizer provides for my project is its MySQL database. This database is fully accessible for performing queries outside of the web interface. It automatically performs data aggregation, which was an initial concern due to how much disk space could be consumed by storing NetFlow data. The database structure consists of seven types of tables, each representing an interval length. The flows can be aggregated based on one minute, five minute, thirty minute, two hour, twelve hour, one day, and one week time intervals. As flows age to the next time interval, they are "rolled up" into that table. For example, when a flow is first received, it is stored in the one minute table. After five minutes has passed, the flow is aggregated into the five minute interval table, and so on. How long the flows are stored can be configured within Scrutinizer.

Device Identification

The NetFlow data of interest for the purposes of my project are the source or destination device, the amount of data transmitted, and the timestamp. Administration of devices on the ResNet network is based on MAC address because it is the only constant identifier. By default, NetFlow identifies a device based on the source or destination IP address. Identification by MAC address is possible with the custom template feature provided in NetFlow version 9. Custom templates allow one to specify which data fields should be exported to the NetFlow collector. Although the UNI network switches support NetFlow version 9, none of the four predefined templates include MAC address fields.

Research of how to create my own template resulted in the conclusion that only templates provided by the network switch vendor can be used. A support case was opened up with the network switch vendor to determine how to export MAC addresses. This feature is supposed to be available in a future firmware upgrade, but so far it has not been released. Without being able to export MAC addresses, my only choice was to use the IP address as the device identifier. The problem with using an IP address to identify a device on the ResNet network is that the device's IP address changes. In the previous ResNet registration system, each device was assigned a unique static IP address.

The new Network Access Control (NAC) system, which will be deployed for the entire ResNet network next year, assigns IP addresses dynamically. This means that a device may be assigned an IP address for an arbitrary length of time, and the IP address may be reassigned to a different device after the lease has expired. To use dynamic IP addresses, I needed some way of getting all IP address leases for a user within a certain timeframe. Dynamic Host Configuration Protocol (DHCP) is a protocol used to assign IP addresses to devices connected to the network. With the assistance of the ResNet network administrator, I discovered the NAC system keeps logs of all DHCP leases, which is exactly what I needed. These logs include the username, MAC address, assigned IP address, and the time the lease was assigned. To provide my application with access to this information within reasonable time, I decided to store the content of the lease log files in a MySQL database.

Between loading and querying the IP lease database and querying the Scrutinizer database, my application requires a lot of database interaction. I researched Java database frameworks and chose to use Hibernate, an object/relational mapping framework. In Hibernate, each table can be mapped to a Java class. Rather than referencing tables directly when performing a query, the mapped class can be used instead. Each query result is an instance of the class, which can be stored and used later in the application. Database connections and table mappings are configured in XML files outside of the application code. A benefit of using Hibernate is it does not require long SQL statements to be embedded in the code. Hibernate provides HQL, a query language more simplified and intuitive than SQL. An alternative to HQL is the Criteria feature, which replaces query language statements with methods that apply logical operators to a query.

Loading the IP lease database consisted of parsing each line of the tab-delimited log files and inserting the data into its respective column. The log files do not include a lease expiration time, which is an important piece of information. To address this issue, I added a lease end column to the database structure. I update the lease end field after each lease is inserted to reflect the change in IP lease assignments.

The IP lease database serves as the basis for auditing bandwidth usage in my application. Each lease represents an interval of time where bandwidth could have been consumed. I perform a query to retrieve all leases assigned to a user within a time interval. The interval is calculated based on the current time and the user's desired timeframe, which can be a day, week, or month. The list of returned result objects is then used to query the Scrutinizer database.

NetFlow Data Retrieval

For each lease retrieved from the IP lease database, I perform a SQL query to get the names of all the five minute interval tables. From the returned tables, I select only those with a date range within the start and end time of the IP lease. For each table, I retrieve all flows where the source IP address or destination IP address equals the lease IP address, and the flow's interval timestamp is within my desired time interval. This query must be performed on several databases because Scrutinizer creates a database for each switch exporting flows. The reason I use the five minute interval tables is because the IP address may change frequently. Use of a longer interval table could result in bandwidth usage being attributed to the wrong device.

A challenge I encountered when configuring Hibernate for the Scrutinizer database is it requires a table name to be defined in the mapping file. Because I have a multitude of tables and the table names are dynamic, it is not feasible for me to create a mapping file for each table. The solution I found to this problem was to set the table name in the mapping file to something arbitrary, then override it later before the query is executed.

Each flow returned from the query is stored in a device object. This device object contains the device's MAC address and its bandwidth usage. The bandwidth usage is represented by two objects, one for incoming flow data (ingress), and the other for outgoing flow data (egress). Each of these objects contains a collection of timestamps and their corresponding bytes of data transmitted. These device objects are then passed to the web interface to be presented to the user.

Web Interface

In order to calculate the bandwidth usage, my application requires a username and timeframe. These pieces of information are retrieved from the web interface. I used Java Server Faces (JSF) to integrate the Java backend and web interface. As a model-view-controller framework, JSF provides support for separation of server-side business logic from the client-side web interface. JSF uses "backing beans" to store information retrieved from the web interface. A backing bean is a Java class with methods to get and set the data it stores. It can also contain action methods mapped to JSF elements, such as buttons. When the form submit button is clicked in my web application, an action method is called to compute the bandwidth usage for the username and timeframe stored in the backing bean.

Finding a way to display the bandwidth usage data in a graph was a difficult task. Originally I planned to use Google Chart, which generates a chart based on parameters defined in an embedded URL. After researching this library, I decided not to use it because it is still in beta and the URL parameters are not intuitive. I researched graphing libraries built specifically for JSF and found two promising options, PrimeFaces and OpenFaces. The major flaw with both of these libraries is that they do not generate axis tickmarks based on the dataset. They generated a tickmark and label for every interval timestamp in my data set, which resulted in the graph being unreadable. To fix this issue, I aggregated data points within certain time intervals. While this solution worked, it was complicated and it did not feel right to manipulate the raw data points to make them fit on a graph.

After digging for more graphing libraries, I stumbled upon JSFlot, a JavaScript-based JSF charting tool. Building the chart was relatively straightforward and the library provides a lot of customization options, including a time mode option. This option recognizes the x-coordinate as a timestamp representing the number of milliseconds since the epoch (January 1, 1970). A major advantage of this library over PrimeFaces and OpenFaces is that it automatically generates axis tickmarks based on the values in the data set. The only issue I ran into with JSFlot is it assumes the x-coordinate timestamps are in Greenwich Mean Time (GMT). I could not find a timezone setting, so I had to convert the data points so the displayed GMT value would be the same as the desired Central Standard Time (CST) value.

The last step in developing my web application was authentication. Authentication is important in my application because only authorized users should be able to access the administrative interface, and non-administrative users must be identified in order to audit their own bandwidth usage. The authentication mechanism I chose to use for this project is the Jasig Central Authentication System (CAS). CAS has become the standard at UNI for authenticating with CatID. In order to use CAS, I sent a request to ITS-Information Services to have my application URL granted access to the CAS server. I configured my web application to use CAS by defining a series of CAS filters in the web configuration file. These filters essentially redirect the user to a CAS login if they have not been authenticated or if their authentication has expired (see Appendix A).

One of the downfalls to CAS is it does not have a built-in way to determine if a user is an administrator. To distinguish between administrators and regular users, I store names of authorized administrators in a file. If the username provided by CAS is found in the file, a field allowing an administrator to enter a username is rendered. If the user is not an administrator, this field is not rendered.

Description of Final Work

The final form of my project is a web application with an interface for administrators and an interface for students registered for ResNet access. Administrators are able to enter the username of a student and a timeframe and view the student's bandwidth usage during that timeframe (see Appendix B). Similarly, a student registered for ResNet access may choose a desired timeframe and view their own bandwidth usage during the selected timeframe (see Appendix C). A chart is generated for each device registered to the student, as well as a total chart combining bandwidth usage for all of the student's devices (see Appendix D through Appendix F). Each chart separates bandwidth usage into ingress bytes and egress bytes. The amount of ingress, egress, and total bandwidth usage corresponding to each chart is also displayed.

In its current state, this application is a proof of concept. It uses a test database containing a week's worth of data gathered in November from a trial version of Scrutinizer. In order to get results from my databases, I must use a date in November as the current date. ITS-Network Services is planning to purchase a license for Scrutinizer, so I hope to get my project moved to a production version before I graduate. My application only works for students registered on the NAC system, which currently includes students living in Rider Hall and College Courts. Next year all students will be on NAC, so this will not be an issue in the near future.

The difficulty I encountered when attempting to tie bandwidth usage to a device resulted in a limitation to the scope of my project. An intended feature was to display a list of the top bandwidth users in the administrative interface. Using an IP address as the device identifier made this feature infeasible. For each device registered on ResNet, my application would need to retrieve all leases assigned to that device within the selected timeframe. For each lease, it would then need to retrieve all flows generated by the device. The dynamic nature of the IP addresses on ResNet required me to use tables with a small interval, which requires a significant number of tables to be queried to cover the entire timeframe. After computing bandwidth usage for every device, my application would then need to determine the top results. With approximately 6,500 devices registered for ResNet access this year, this feature would be computationally expensive and not something that could be loaded in a web application within reasonable time.

While identification by IP address works for my project, this application should be modified to identify devices by MAC address once the network hardware supports it. This would speed up computation by allowing larger interval tables to be used, which would also allow the list of top users feature to be implemented. This application's primary function is reporting bandwidth usage, but it could be expanded in the future to also enforce bandwidth policies. It could determine which users are over a particular threshold and perform an action, such as sending an email to an administrator or restricting the user's Internet access.

Reflection

The work performed for my thesis project will benefit ResNet and ITS-Network Services by providing a proof of concept for using NetFlow to monitor bandwidth consumption. This software serves as the basis for enforcing bandwidth usage policies on ResNet. The application I have produced will be available for future modification and enhancement after I graduate.

Another unintended, but important benefit my project provides is the IP lease database. When viruses are detected or copyright violations are received, the offending device is identified by an IP address. With ResNet transitioning to dynamic IP addresses, determining which device should be restricted access will become a difficult task. The IP lease database will allow network administrators to determine which device was leased an IP address at a specific time.

This project allowed me to build software of a larger scale than what I have had an opportunity to do in most of my classes. I encountered a lot of issues throughout this project, which made me better at problem solving and researching alternative solutions. This project further developed my knowledge of web applications and other technologies I will be using after graduation.

Bibliography

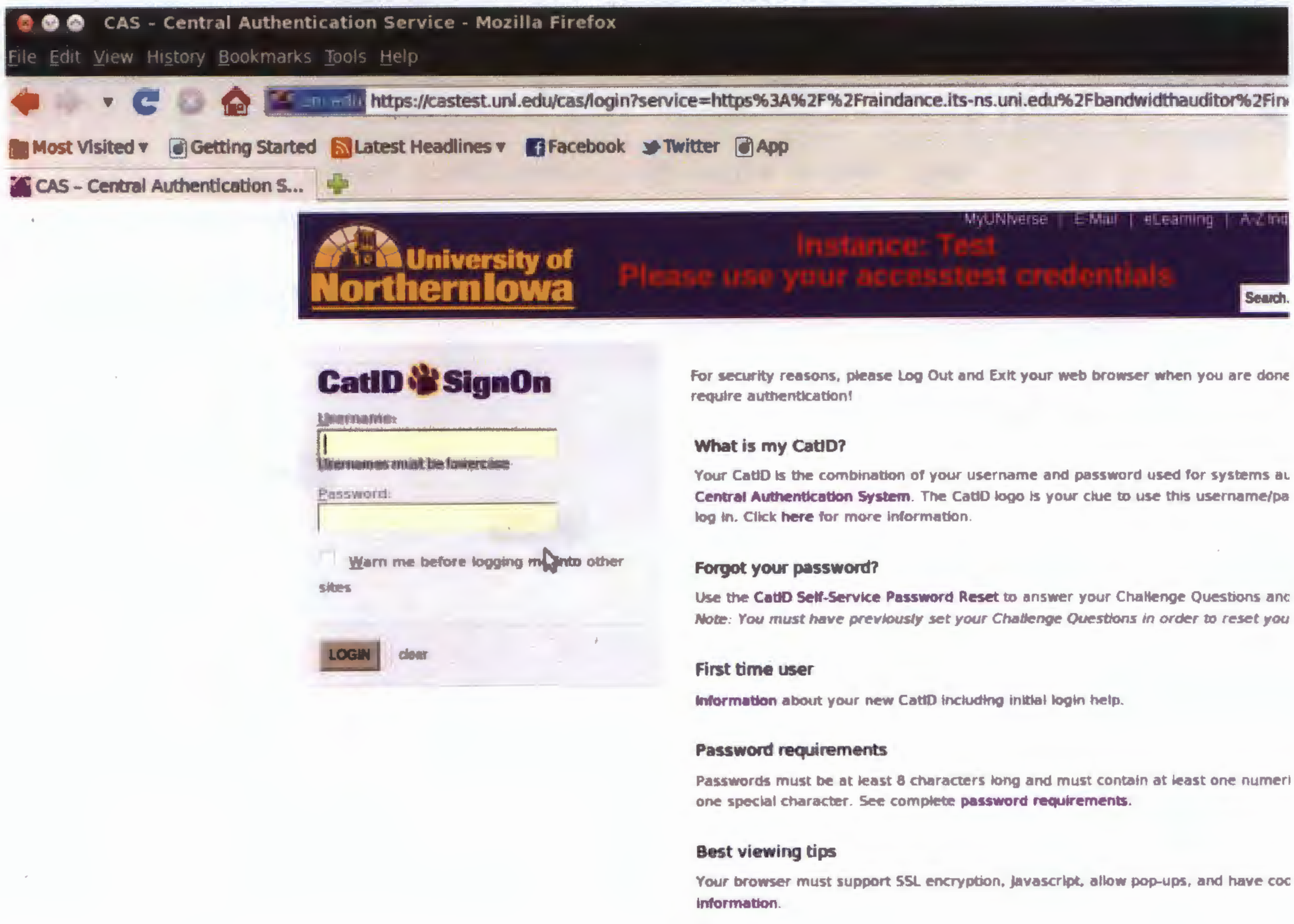
Enterasys Networks. (2010). *Configuring NetFlow.* Retrieved from http://secure.enterasys.com/support/manuals/hardware/NetflowFeatGde071510.pdf

Peterson, L., & Davie, B. (2007). *Computer Networks: A Systems Approach* (4th ed.). San Francisco: Morgan Kaufman Publishers.

Plixer. (2010). *Use a NetFlow & sFlow traffic analyzer to help solve network problems.* Retrieved October 24, 2010, from http://www.plixer.com/products/netflow-sflow/scrutinizer-netflow-sflow.php

Appendix A

CatID Authentication


CAS - Central Authentication Service - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Most Visited Getting Started Latest Headlines Facebook Twitter App
CAS - Central Authentication S...
MyUNIverse | E-Mail | eLearning | A-Z Ind
University of Northern Iowa
Search
CatID SignOn
Username:
Password:
LOGIN
clear

https://castest.uni.edu/cas/login?service=https%3A%2F%2Fraindance.its-ns.uni.edu%2Fbandwidthauditor%2Fin

Instance: Test
Please use your accesstest credentials

Usernames must be lowercase

Warn me before logging me into other sites

For security reasons, please Log Out and Exit your web browser when you are done
require authentication!

What is my CatID?

Your CatID is the combination of your username and password used for systems au
Central Authentication System. The CatID logo is your clue to use this username/pa
log in. Click **here** for more information.

Forgot your password?

Use the **CatID Self-Service Password Reset** to answer your Challenge Questions anc
Note: You must have previously set your Challenge Questions in order to reset you

First time user

Information about your new CatID including initial login help.

Password requirements

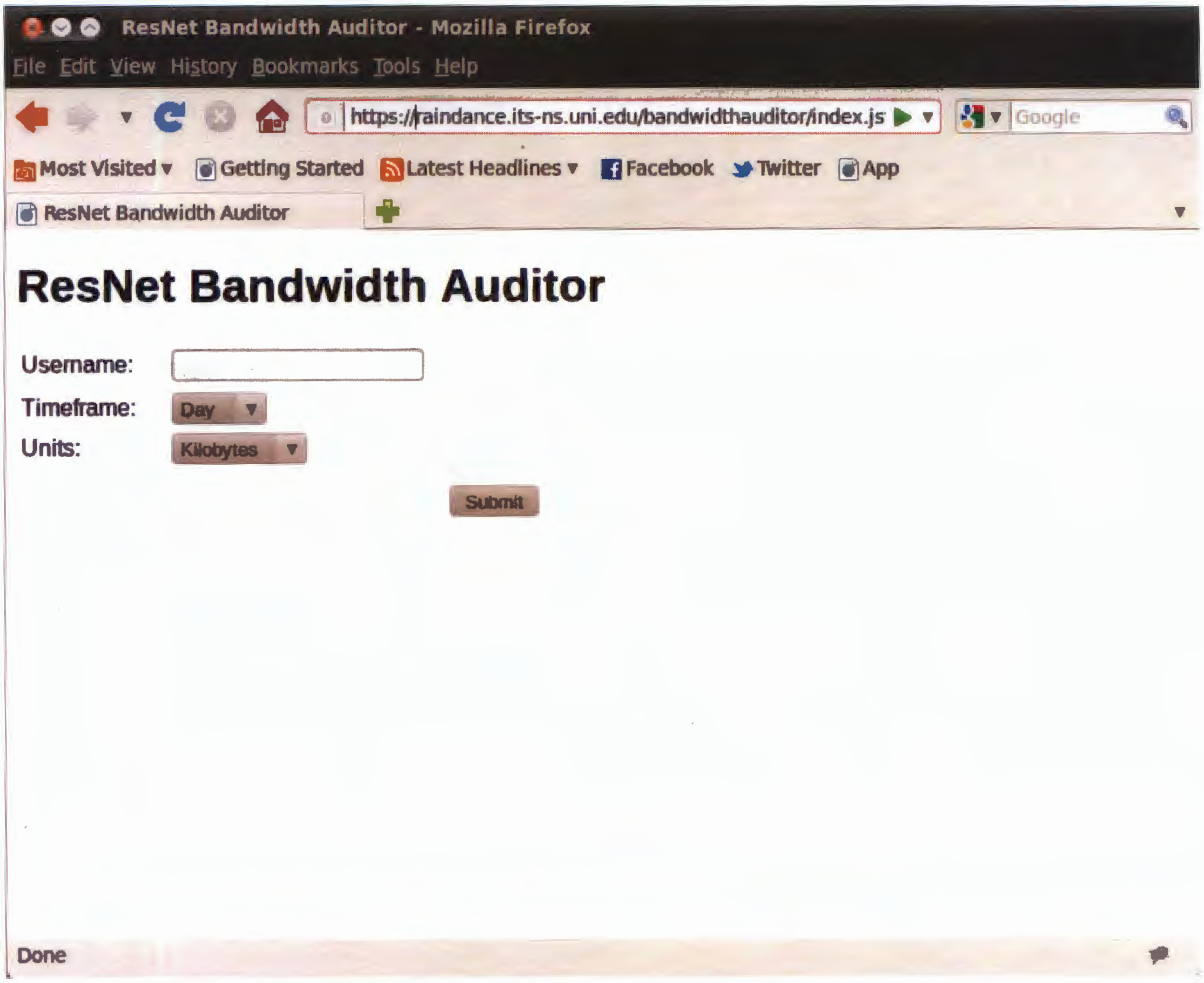
Passwords must be at least 8 characters long and must contain at least one numeri
one special character. See complete **password requirements**.

Best viewing tips

Your browser must support SSL encryption, Javascript, allow pop-ups, and have coc
information.

Appendix B

Administrator Interface


ResNet Bandwidth Auditor - Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://raindance.its-ns.uni.edu/bandwidthauditor/index.js
Google
Most Visited
Getting Started
Latest Headlines
Facebook
Twitter
App
ResNet Bandwidth Auditor
Submit
Done

ResNet Bandwidth Auditor

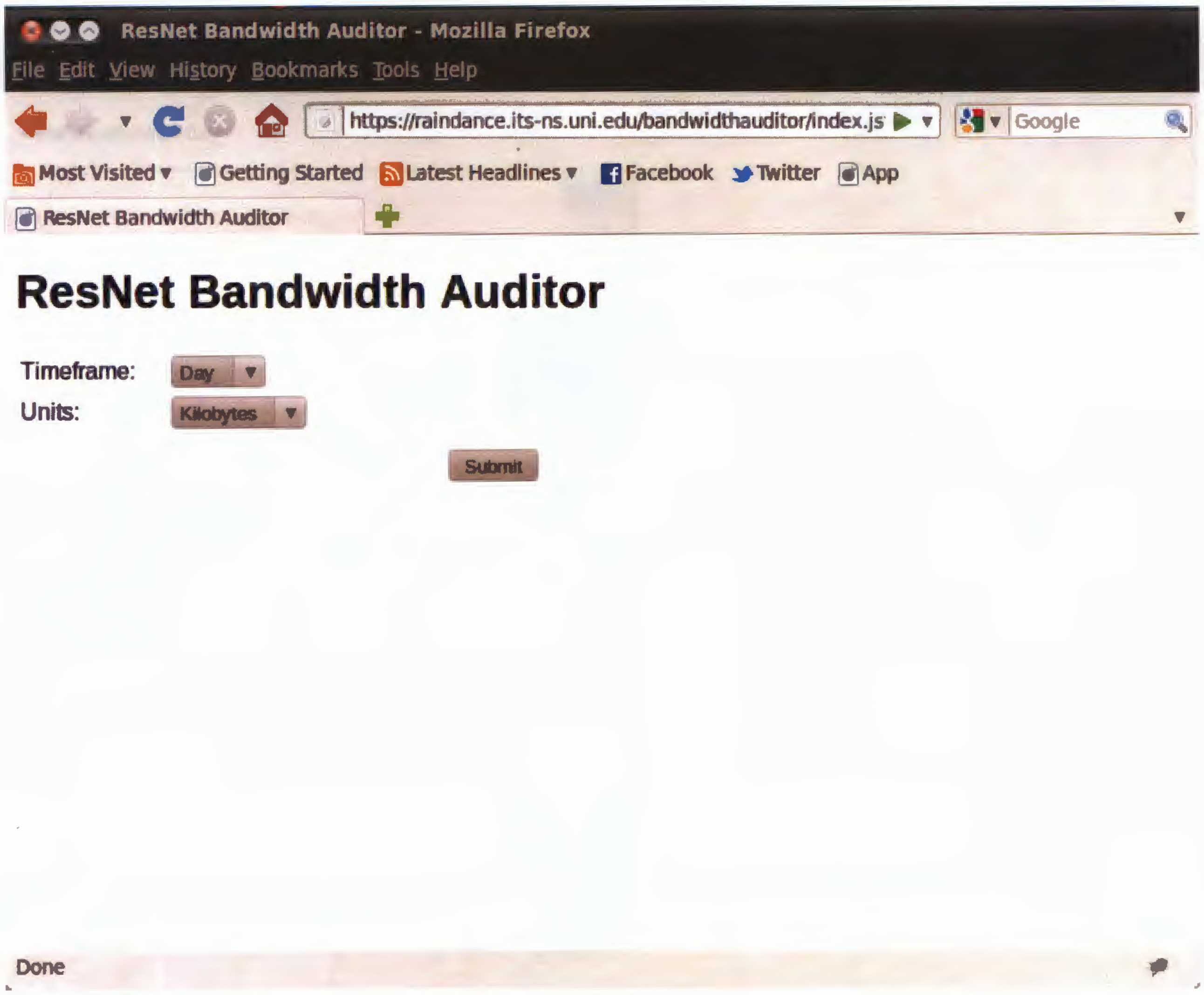
Username:

Timeframe: Day

Units: Kilobytes

Appendix C

ResNet User Interface


ResNet Bandwidth Auditor - Mozilla Firefox

File Edit View History Bookmarks Tools Help

https://raindance.its-ns.uni.edu/bandwidthauditor/index.js

Most Visited Getting Started Latest Headlines Facebook Twitter App

ResNet Bandwidth Auditor

ResNet Bandwidth Auditor

Timeframe: Day

Units: Kilobytes

Submit

Done

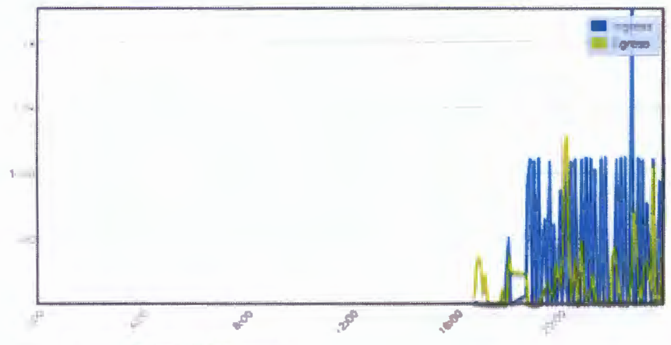
Appendix D

Bandwidth Usage - Day Timeframe

ResNet Bandwidth Auditor

Bandwidth usage for janedoe

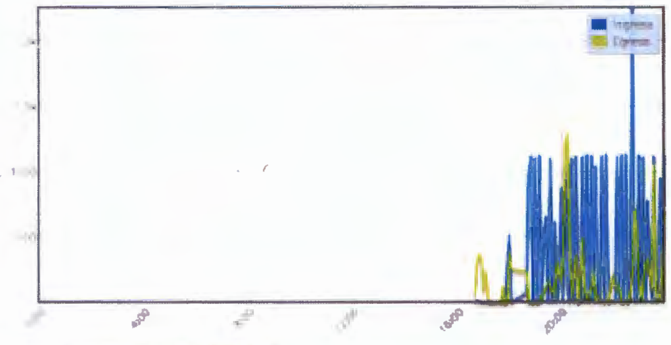
Total bandwidth: day



Ingress: 39259.31 kilobytes
Egress: 26817.08 kilobytes
Total: 66076.39 kilobytes

By device: day

88:AE:1D:3D:C4:16



Ingress: 39259.31 kilobytes
Egress: 26316.0 kilobytes
Total: 65575.31 kilobytes

B8:AC:6F:77:B3:1C

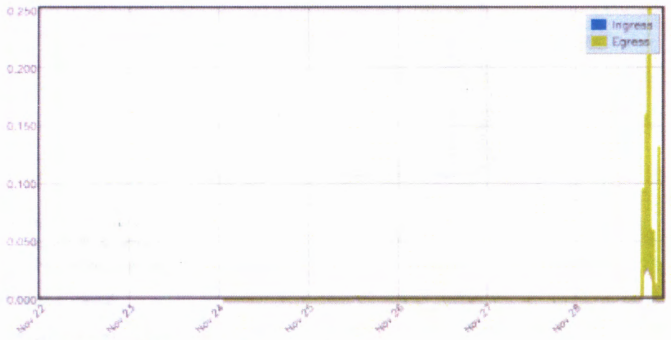
Appendix E

Bandwidth Usage – Week Timeframe

ResNet Bandwidth Auditor

Bandwidth usage for johndoe

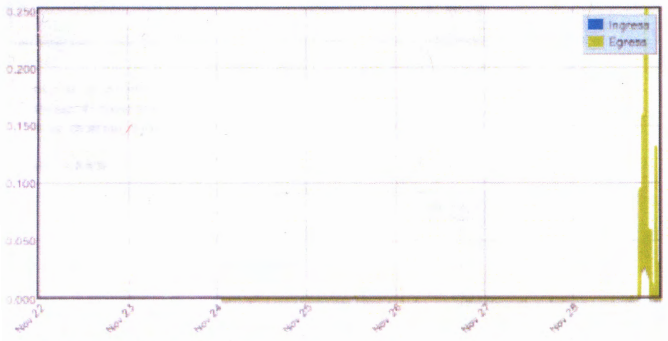
Total bandwidth: week



Ingress: 0.0 megabytes
Egress: 2.22 megabytes
Total: 2.22 megabytes

By device: week

00:23:DF:F8:FA:72

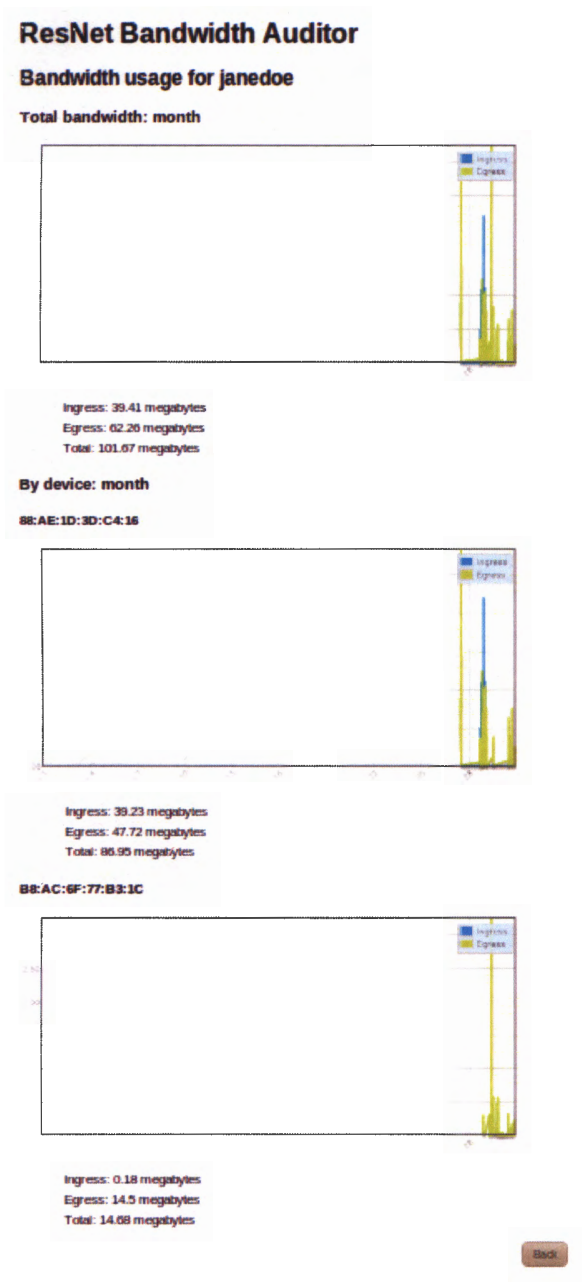


Ingress: 0.0 megabytes
Egress: 2.22 megabytes
Total: 2.22 megabytes

Back

Appendix F

Bandwidth Usage – Month Timeframe



ResNet Bandwidth Auditor

Bandwidth usage for janedoe

Total bandwidth: month

Ingress: 39.41 megabytes
Egress: 62.26 megabytes
Total: 101.67 megabytes

By device: month

88:AE:1D:3D:C4:16

Ingress: 39.23 megabytes
Egress: 47.72 megabytes
Total: 86.95 megabytes

B8:AC:6F:77:B3:1C

Ingress: 0.18 megabytes
Egress: 14.5 megabytes
Total: 14.68 megabytes