Time-based Bandwidth Profiles for Campus Traffic Control

Michael Scott
michael_scott@college.harvard.edu

Zennie Wey
zennie_wey@college.harvard.edu

***Abstract*—Increasing amounts of Wi-Fi usage on Harvard's campus necessitates a novel approach to traffic control. Students have long been reporting dropped connections and inconsistent network performance to HUIT, who have been hard at work updating an aged and unsuitable infrastructure. Barring the extremely expensive and time critical solution of mass renovations to expand coverage and bandwidth, a more economically feasible option exists: traffic control to properly and fairly allocate bandwidth at times of high usage and prevent wasteful allocation at times of low usage. This system can reasonably be implemented using software-defined networking to control link bandwidth based on two factors: building profile (a building's access point usage pattern) and time.**

***Index Terms*—Wi-Fi, Bandwidth, Mininet, SDN**

I. Introduction

The proliferation of IoT devices and the growing number of connected devices per person has resulted in an increased need for fast internet speeds. At Harvard, students arrive with, one average, four Wi-Fi connected devices, causing higher demand for wireless coverage. This has impacted the stability of the wireless network. As a result, HUIT has made improving the wireless experience a top 5 goal for the upcoming year.

Software defined networking allows for network flexibility without physical infrastructure change. Services (such as bandwidth) can be relocated to areas of need. Because students aggregate in areas such as the Science Center and Maxwell Dworkin during the day and the houses at night, it is pertinent that there is sufficient bandwidth to support the needs of the students.

Because usage is dependant on time and place, it is unreasonable to employ an even distribution of bandwidth throughout an area. We want to be able to dynamically allocate bandwidth to areas of high use. This allows for load balancing, especially during peak periods, which could ideally aid in improving the wireless experience for students. As the campus and number of wireless devices grows, an unchanging infrastructure will simply result in bottlenecks, and through SDN, we hope to solve this issue.

Prior work in this area has been done in relation to cloud networks and bandwidth guarantees in order to predict bandwidth demands. In this way, the provider can present the client with a guarantee that matches his or her usage based on past history. The guarantees no longer need to be static and can instead be a "difference guarantee" [1].

II. Proposed Approach

We chose to take on this problem because it is a prevalent issue for not only students but also faculty and staff. Revamping the physical infrastructure of the current system - upgrading and adding access points in addition to changing the wiring - is a significant undertaking. The process is gradual and the task continual. Software defined networking allows for an intermediary in between phasing out the old system and fully upgrading the campus Wi-Fi connectivity. It's been extensively explored in the programming assignment, and is a cost effective way to boost connectivity at peak hours.

HUIT is overseeing a steady transition from older technologies to newer, more adaptive ones. The current system is a Cisco-based infrastructure supplemented with SecureW2 for network security, but future plans include using Aruba to adapt to a more mobile-connected campus population and alleviate issues with access point roaming, in addition to adapting the Protected Extensible Authentication Protocol as a security standard. This process will naturally take time to complete, during which the campus network is still under heavy use.

As a more immediate, low-investment solution to bandwidth limitations on campus, software defined networking can be used to intelligently distribute bandwidth to high-traffic access points and limit excess bandwidth being spent on comparably inactive access points.

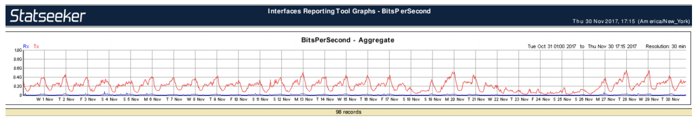


Fig. 1. Access point usage data from Leverett House, a residential building. Note pattern of cyclic peaks and valleys.

Upon analyzing graphs containing the bits per second aggregate, it is evident that we are able to define certain times at which the internet usage is at a high. These trends allow us to define certain hours as "problematic". Due to the consistency of the trends, there is less of a need for machine learning and instead, bandwidth can be allocated with respect to time.

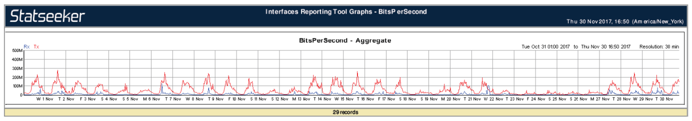


Fig. 2. Access point usage data from Maxwell Dworkin, a STEM lab building. Note the more gradual usage transitions and the secondary peak compared to Fig. 1.

The goal is to implement a program in Python that can dynamically allocate bandwidth depending on time of day and traffic. The program should have the capability of changing link bandwidths on the fly. This should be done without any downtime, seeing as the updates to the network occur multiple times on a daily basis.

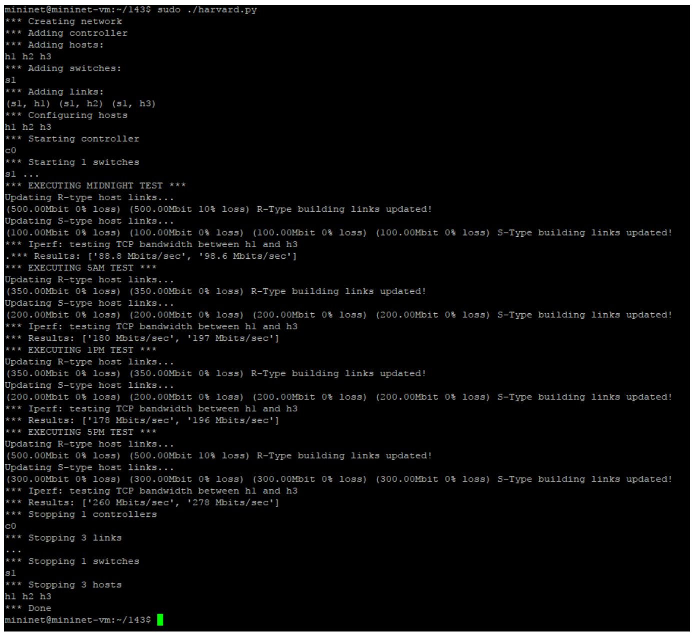
The endeavor will be considered a success if bandwidth test results from before running the algorithm and after reflect general improvement. Since we are testing in a virtual environment, the success will have to be measured given test results. Testing will be done by checking maximum bandwidth of links at various times of the day. If the results match the expected maximums based on their profiles, the algorithm is successful.

III. Setups for Experiment and Analysis

The algorithm is contained within a Python 2.7.6 script that runs a Mininet 2.2.2 virtual network. This network contains three primary nodes, representing the data received from HUIT: Leverett House, Maxwell Dworkin, and Pierce Hall. All other elements of the network are abstracted away to represent only nodes with concrete access point usage data.

The Python script initializes an instance of *HarvardTopo*, a Mininet topology with the three aforementioned hosts and a singular switch with connections to each. Each host is stored in a dictionary as a tuple, containing both the Mininet node itself and a string representing a building's "profile", or its expected pattern of bandwidth consumption. Links do not initially have bandwidth restrictions; control of this is done through the *updateLinks* function, which, when given a building profile to update and a network, will change the bandwidth according to predetermined levels.

Ideal bandwidth is estimated from the graphs of access point usage for each building. From the data currently accessible, two profiles have been created: *'R'*, representing residential hosts like Leverett, and *'S'*, representing STEM laboratories and office buildings such as Maxwell Dworkin and Pierce. *R-Type* hosts are characterized by an alternating downtime and uptime, with downtime beginning at approximately 5 a.m. and uptime beginning at 5 p.m. These patterns coincide



```
mininet@mininet-vm:~/143$ sudo ./harvard.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(s1, h1) (s1, h2) (s1, h3)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** EXECUTING MIDNIGHT TEST ***
Updating R-type host links...
(500.00Mbit 0% loss) (500.00Mbit 10% loss) R-Type building links updated!
Updating S-type host links...
(100.00Mbit 0% loss) (100.00Mbit 0% loss) (100.00Mbit 0% loss) (100.00Mbit 0% loss) S-Type building links updated!
*** Iperf: testing TCP bandwidth between h1 and h3
.*** Results: ['88.8 Mbits/sec', '98.6 Mbits/sec']
*** EXECUTING 5AM TEST ***
Updating R-type host links...
(350.00Mbit 0% loss) (350.00Mbit 0% loss) R-Type building links updated!
Updating S-type host links...
(200.00Mbit 0% loss) (200.00Mbit 0% loss) (200.00Mbit 0% loss) (200.00Mbit 0% loss) S-Type building links updated!
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['180 Mbits/sec', '197 Mbits/sec']
*** EXECUTING 1PM TEST ***
Updating R-type host links...
(350.00Mbit 0% loss) (350.00Mbit 0% loss) R-Type building links updated!
Updating S-type host links...
(200.00Mbit 0% loss) (200.00Mbit 0% loss) (200.00Mbit 0% loss) (200.00Mbit 0% loss) S-Type building links updated!
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['178 Mbits/sec', '196 Mbits/sec']
*** EXECUTING 5PM TEST ***
Updating R-type host links...
(500.00Mbit 0% loss) (500.00Mbit 10% loss) R-Type building links updated!
Updating S-type host links...
(300.00Mbit 0% loss) (300.00Mbit 0% loss) (300.00Mbit 0% loss) (300.00Mbit 0% loss) S-Type building links updated!
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['260 Mbits/sec', '278 Mbits/sec']
*** Stopping 1 controllers
c0
*** Stopping 3 links
...
*** Stopping 1 switches
s1
*** Stopping 3 hosts
h1 h2 h3
*** Done
mininet@mininet-vm:~/143$ 
```

Fig. 3. Example output of the *updateTest* function.

with student population present in dormitories on a standard day, and are reflected in the Python script by a bandwidth of 350 Mbits from 5 a.m. to 5 p.m. and a 500 Mbit connection from 5 p.m. to 5 a.m..

S-Type hosts demonstrate a gradual, double-peaked usage pattern. Their allocation profile divides the usage into four phases:

1) 10 p.m. - 5 a.m.: Downtime, 100 Mbit link bandwidth.
2) 5 a.m. - 1 p.m.: Initial peak, 200 Mbit link bandwidth.
3) 1 p.m. - 5 p.m.: Primary peak, 300 Mbit link bandwidth.
4) 5 p.m. - 10 p.m.: Descent, 125 Mbit link bandwidth.

Links are updated immediately upon starting the network, allowing the network to be restarted or modified at any time and still maintain accurate bandwidth profiles. Continual updates are scheduled using the Python *schedule* library, running *updateLinks* at the beginning of each phase. The network can otherwise be controlled using Mininet's command line interface for all other purposes while scheduled link updating occurs in the background.

IV. Results

Testing of the algorithm is done via an *updateTest* function, which establishes several *datetime.datetime* objects representing the various times of day that encompass all of the possible timespans that link updates would change between. A bandwidth test, performed using Mininet's *iperf* function, is executed between each network update, which tests the bandwidth between the first host and the final host if not given any arguments. In the case of *HarvardTopo*, the test is performed between Leverett House ('*h1*') and Pierce Hall ('*h3*'). The command-line results of *updateTest* are shown below in Fig. 3.

Testing reveals that the algorithm functions exactly as intended. At midnight, the bandwidth between Leverett and Pierce reaches a max of 98.6 MBits/sec. This result is expected, as Pierce (an S-type building) has a maximum bandwidth of 100 MBits/sec from 10 p.m. to 5 a.m.. For the remainder of testing results (shown in the table in Fig. 4), the bandwidth follows a similar pattern of capping at the smaller host's link size.

Time	Iperf Server	Iperf Client	Expected Cap
0:00	89.5 Mbits/sec	99.0 Mbits/sec	100 Mbits/sec
5:00	178 Mbits/sec	196 Mbits/sec	200 Mbits/sec
13:00	260 Mbits/sec	277 Mbits/sec	300 Mbits/sec
17:00	111 Mbits/sec	123 Mbits/sec	125 Mbits/sec

Fig. 4. Results of an iperf test through the *updateTest* function.

Seeing as this algorithm was capable of accurately updating link bandwidths without any network downtime, we deem this successful.

V. Conclusion and Future Work

Moving forward, given more time and resources, it would perhaps be useful to implement an algorithm incorporating machine learning. Due to the nature of student life, the internet usage schedule is easy to predict - on weekdays, students go to class during the day and return during the mid afternoon. On weekends, students are more likely to remain concentrated around the houses. The users, independent of class year and affiliation, all share similar habits. If our algorithm were extended to a more public setting, however, we would be unable to adapt to new changes as it is specifically tailored to the habits of Harvard students. Machine learning allows for a broadened scope. Pattern recognition makes independent adaptation to a changing environment possible. Not only would improve a public network, but the data collected, if in for example, a shopping area, can also be used to predict trends shopping habits as well as restaurant choice, depending on highest concentration of bandwidth usage. However, due to the lack of training data available, machine learning was not feasible for the scope of this project.

Additional proposed features include a ”bandwidth bump” mechanism, accounting for weekends and vacations, and additional profiles. One issue that occurs a result of a predetermined bandwidth cap is occurrences of abnormally high bandwidth usage. These can be due to a variety of reasons, such as events that draw a large number of visitors to campus or reading period. A system capable of ”bumping up” the bandwidth, or slightly increasing the link size whenever a certain threshold of packet loss occurs (much like TCP's Slow Start) would remedy this situation.

In addition to high-traffic events, there are also periods of low traffic caused by university holidays and breaks, and to a lesser extent weekends in certain building profiles. Examples include Leverett House AP usage (Fig 1.) shows a distinct decline in usage pattern from November 22 to November 27, coinciding with Thanksgiving break. Predicting these downtimes and adding exceptions to the algorithm would prevent a drastic difference between bandwidth cap and actual usage.

Finally, an obvious expansion of this project would include the continual addition of profiles to suit the needs of the entire campus. It is highly unlikely that all buildings on campus share the same access patterns. Libraries, dining halls, lecture halls and classrooms, and athletic facilities are all likely candidates for new profiles to best match their usage. This naturally requires data that is currently unavailable and would necessitate additional cooperation with HUIT. However, with access to this data, adding new profiles is a simple process that needs only be done once for a variety of buildings, assuming data patterns are consistent.

Despite so, the experience of this project speaks to the value of critical analysis of problems before addressing them. The intuitive algorithmic solution to a problem like this likely involved some form of machine learning, using cluster analysis to predict access point usage throughout the day and year in order to create a best-fit solution. However, examining the data and thinking critically reveals that no such process is necessary because data usage is predictable based exclusively on the characteristics of a building and time. Necessity breeds innovation; since the data necessary to build a machine learning solution was unobtainable in this circumstance due to data release restrictions and the inability to set up network monitoring tools over a long period of time, the profile-based solution went from a reasonable possibility to the best one.

References

[1] K. LaCurts, J. C. Mogul, H. Balakrishnan, and Y. Turner, “Cicada: Introducing predictive guarantees for cloud networks,” 2014.