AN APPROACH TO SHAPING AWAY WIRELESS INTERFERENCE IN 802.11 TRAFFIC AT DIFFERENT TRANSMISSION RATES

by

Jimmy Wong

B.Sc., The University of British Columbia, 2006

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

September 2011

© Jimmy Wong, 2011
Abstract

IEEE 802.11 wireless networks are widely deployed and used nowadays, especially in enterprise and university settings. The widespread usage means increased airspace congestion when many users connect at the same time, which slows down the wireless network performance. Two infamous problematic scenarios are the hidden terminal and the exposed terminal, where a victim node will suffer decreased throughput because of interference from other nodes which are unfairly using up the available airspace. This wireless unfairness occurs due to the design of the 802.11 protocol and the vast number of clients connected.

The thesis presents a VOIDShaper Engine, which captures the network traffic at a central upstream router through which all wireless access points connect, and uses a previously developed VOID tool to analyze the traffic and generate a network interference map. With this map, VOIDShaper can find the interference between pairs of TCP Flows, and apply corrective traffic shaping rules on the interfering TCP flows directly at the router. All experiments were run in the Emulab testbed; we see that with traffic control, bandwidth fairness is achieved by using the HTB and SFQ queuing strategies at the router.

The goals of this work are 1) to use VOIDShaper to detect and shape wireless traffic at the same 802.11 transmission rates due to interference, 2) to analyze how VOIDShaper responds to wireless interference when the nodes are transmitting at different wireless data rates, 3) to devise an algorithm and shaping policy for traffic which is sent at symmetric or different 802.11 transmission rates.
Table of Contents

Abstract ......................................................................................................................... ii

Table of Contents ........................................................................................................ iv

List of Tables .................................................................................................................... v

List of Figures .................................................................................................................... vi

List of Symbols, Abbreviations or Other ................................................................. vii

Acknowledgements ......................................................................................................... viii

Chapter 1: Introduction .................................................................................................. 1
  1.1 Background on 802.11 .......................................................................................... 2
  1.2 Wireless unfairness ............................................................................................... 3
  1.3 Problematic scenarios ........................................................................................... 4
     1.3.1 Hidden terminal ............................................................................................... 4
     1.3.2 Exposed terminal ............................................................................................ 5
  1.4 Thesis outline ......................................................................................................... 5

Chapter 2: Related work ................................................................................................. 7
  2.1 Mapping out the network ....................................................................................... 7
  2.2 Solutions to wireless fairness ............................................................................... 8
  2.3 Changing the 802.11 protocol ............................................................................. 9

Chapter 3: The VOIDShaper engine ............................................................................ 10
  3.1 Assumptions .......................................................................................................... 10
  3.2 Defining wireless fairness ..................................................................................... 11
  3.3 Shaping rate/fairness determination ................................................................... 11
     3.3.1 Two wireless nodes sending at the same txrate ............................................. 12
     3.3.2 Two wireless nodes sending at different txrates .......................................... 12
  3.4 Topology/environment ......................................................................................... 13
  3.5 VOIDShaper details .............................................................................................. 15
     3.5.1 Network capture ............................................................................................. 15
     3.5.2 Capture analysis ............................................................................................. 16
     3.5.3 VOID analysis .............................................................................................. 16
3.5.3.1 Throughput function and the coefficients .............................................................. 16
3.5.3.2 Sample VOID xml output ...................................................................................... 17
3.5.3.3 Visualizing the network traffic .............................................................................. 18
3.5.4 Traffic shaping engine (shaper) ...................................................................................... 19
3.5.4.1 Grouping the interfering nodes for shaping ........................................................... 19
3.5.4.2 Shaping the TCP flows together ............................................................................ 19
3.5.4.3 An alternative: shaping the flows individually ...................................................... 20
3.6 Proposed traffic shaping algorithm .................................................................................... 21
3.7 When to remove the shaping rules? ................................................................................... 23

Chapter 4: Evaluation .......................................................................................................... 24
  4.1 Experiment setup ............................................................................................................... 25
  4.1.1 Wireless node selection .................................................................................................. 26
  4.1.2 Emulab challenges .......................................................................................................... 26
  4.2 Interfering traffic at symmetric transmission rates (36Mbps) ............................................. 27
  4.3 Alternative shaping method: shaping each flow individually ............................................ 32
  4.4 Interfering traffic at symmetric transmission rates (11Mbps), fair sharing ......................... 34
  4.5 Interfering traffic at different txrates, with strong sender at slower rate (5.5Mbps) and weak sender at faster rate (36Mbps) ................................................................................................ 35
  4.6 Interfering traffic at different txrates, with strong sender at faster rate (36Mbps) and weak sender at slower rate (5.5Mbps) ........................................................................................................ 37
  4.7 Summary ............................................................................................................................. 41

Chapter 5: Conclusion and future work ............................................................................. 43
  5.1 Conclusion .................................................................................................................... 43
  5.2 Limitations and future work ........................................................................................... 43

Bibliography .......................................................................................................................... 45

Appendices ............................................................................................................................. 49
  Appendix A ...................................................................................................................................... 49
  Appendix B ...................................................................................................................................... 50
  Appendix C ...................................................................................................................................... 51
# List of Tables

Table 4.1  Traffic shaping and VOID analysis results for shaping flows together ............... 28
Table 4.2  Traffic shaping and VOID analysis results for shaping flows individually ....... 33
Table 4.3  Throughputs and VOID analysis results for 11Mbps wireless traffic ................. 34
Table 4.4  Throughputs and VOID analysis results for wireless traffic at different txrates . 36
Table 4.5  Throughputs and VOID analysis results for wireless traffic at different txrates . 39
List of Figures

Figure 1.1    Diagram of hidden terminal problem ................................................................. 4
Figure 1.2    Diagram of exposed terminal problem ............................................................. 5
Figure 3.1    Hidden terminal topology setup in emulab ......................................................... 13
Figure 3.2    Emulab machine location map ........................................................................... 14
Figure 3.3    The components of the pipeline design ............................................................ 15
Figure 3.4    Plot of network capture data showing interference ........................................ 18
Figure 3.5    Linux traffic shaping queuing structure for shaping flows together ............... 20
Figure 3.6    Linux traffic shaping queuing structure for shaping flows individually .............. 20
Figure 4.1    The effect of shaping rate on throughput of two TCP flows ....................... 29
Figure 4.2    Throughput time series for both flows before shaping ....................................... 31
Figure 4.3    Throughput time series for both flows after shaping ......................................... 32
List of Symbols, Abbreviations or Other

D-ITG [1] – Distributed Internet Traffic Generator
IEEE – Institute of Electrical and Electronics Engineers
TCP – Transmission Control Protocol
IP – Internet Protocol
MAC – Media/Medium Access Control
bps – Bits Per Second
Mbps – Megabits Per Second
Kbps – Kilobits Per Second
MBps – Megabytes Per Second
KBps – Kilobytes Per Second
TP – Throughput
Txrate – Transmission Rate
WLAN – Wireless Local Area Network
Acknowledgements

I offer my gratitude to my supervisor, Dr. Mike Feeley, for offering the much needed guidance and support throughout. The discussions we had were always supportive, inspiring, and fruitful. I am also grateful to Kan Cai for introducing me to this interesting topic, and bearing with all the help needed and questions asked.

Lastly, I want to thank especially my fiancée and my family for their endless support.
Chapter 1: Introduction

IEEE 802.11 wireless networks are widely deployed and used nowadays, especially in enterprise and university settings. At any given time, wireless devices are transmitting and receiving data concurrently. The increased usage means increased congestion when devices try to access the shared airspace - which is the medium radio frequencies work in. Two infamous problematic scenarios are the hidden terminal and the exposed terminal, where a victim node will suffer decreased throughput because of interference from other nodes which are unfairly using up the available airspace. This wireless unfairness occurs due to the design of the 802.11 protocol and the vast number of clients connected.

Previous work by Kan Cai has shown that traffic shaping (Shaper) [5] can be used to effectively control TCP flows such that wireless nodes share bandwidth fairly. And, with VOID [2] (Wireless Online Interference Detection), wireless interference can be correctly inferred from analyzing TCP traffic patterns. In this thesis, VOID will be bridged with Shaper for a more complete solution to alleviating wireless interference; this will be explained in detail in Chapter 3.

VOID or Shaper has only considered wireless nodes sending at a symmetric transmission rate (txrate), such as 36Mbps. This makes traffic shaping trivial because the assumption is that all nodes send at the same txrate. In this work, wireless nodes sending at different txrates are considered, such as 36Mbps versus another node at 5.5Mbps. This is hard because neither VOID or Shaper has information from the access point regarding the txrate of the wireless senders – the proposed solution must analyze the VOID and TCP throughput numbers to decide how to shape the traffic.

The proposed tool is called VOIDShaper, and our goals are,
1) To use VOIDShaper to detect wireless traffic at the same txrates,
2) To analyze how VOIDShaper responds to wireless interference when the nodes are transmitting at different txrates.
3) To propose an algorithm (containing a set of shaping policies) for wireless traffic at symmetric or different txrates, since VOIDShaper does not know the txrate information from the access point.

This work will explore whether VOIDShaper can detect wireless interference at symmetric or different txrates and what the correlation data looks like, what shaping policies should be considered and applied for the symmetric and different txrates, and finally, whether the proposed algorithm can effectively shape away wireless interference.

1.1 Background on 802.11

802.11 is a set of standards created by the IEEE Standards Committee. It defines the protocols for designing and operating 802.11 wireless devices, such as wireless network cards in computers and phones, and wireless access points and routers. The first version was IEEE 802.11, which was introduced in 1997. Later in 1999, IEEE 802.11a and 80211.b were released. In 2003, IEEE 802.11g was introduced after the widespread popularity and adoption of 802.11b. Today in 2011, devices are using the newest 802.11n standard, defined in 2009. [6] Each standard defines a set of data transmission rates; for example 802.11g rates include 1, 2, 5.5, 11, 6, 9, 12, 18, 24, 36, 48, 54Mbps.

Last year in 2010, the Wi-Fi Alliance reported that "Wi-Fi shipments grew to 761 million products, a 29% increase from 2009" [7]. The popularity and commonness of Wi-Fi products is definite nowadays - Wi-Fi chipsets are embedded in many electronic products. In general, we use 802.11 to refer to the devices using any of the IEEE 802.11 standards. In our experiments, we will use 802.11g for testing.

802.11 uses the radio as its medium to carry packets wirelessly. The radio airspace is shared with many other devices which also operate on the frequency. For example, 802.11g uses the 2.4Ghz (industrial, scientific and medical) ISM band for operation, which is shared with other devices especially 2.4Ghz cordless phones and microwaves. Such devices can easily interfere with the 802.11 signals [8], causing signal quality degradation, meaning packet loss.
at the MAC layer. Wireless networks are always susceptible to outside interference, and in this study we are focusing on the interference between 802.11 devices.

In an enterprise or university setup, multiple 802.11 access points are deployed and connected to a wired network backbone. Users have to be physically located sufficiently close to an access point to connect wirelessly using their 802.11 compliant devices (such as laptop, cell phones, tablets, etc). In one university wireless local area network (WLAN) usage study [9], it was found that users are using their wireless connection as the primary one, even though a wired connection was available in the office; also, usage patterns for the wireless connection was no different from the wired (peer-to-peer bandwidth hungry applications were used on WLAN).

In a residential apartment or public hotpot setting, the situation is potentially worse with a high density of individually deployed wireless access points. One study [10] with data on several large US cities has shown that performance of end-clients can suffer significantly as these unmanaged access points are not configured at all to cooperate. In our case, we are concerned with enterprise environments where access points are connected to a central router such that traffic can be controlled.

1.2 Wireless unfairness
The cause of unfairness is airspace congestion [5]. As more 802.11 clients try to send packets wirelessly through the air, the medium becomes filled with more packets than 802.11 can handle. Ideally, wireless nodes cooperate with each other by requesting and using its fair share of the network. However, in an unfair scenario, one node can dominate the airspace excessively such that other nodes cannot get its fair share. This causes the victim nodes to try and recover by retransmitting quickly at the 802.11 MAC layer. The increased packet loss will cause the TCP flow at the higher layer to back off and suffer extremely low throughput.

IEEE 802.11 is designed to support wireless clients in a decentralized fashion. That means each node determines its data transmission rate based on the channel conditions (the
airspace) at a particular instant. A node which picks a lower transmission rate will take more time in transmitting a data frame - in order to reduce the bit error rate and retransmissions due to packet loss. Since the nodes are not coordinated, an individual node will want to maximize its own throughput, which can be unfair to other wireless nodes and/or decrease the aggregate throughput of the system [11].

1.3 Problematic scenarios
Two problematic scenarios can exist in wireless networks creating an unfair usage of the wireless airspace. The first scenario is called a hidden terminal, and the second is called an exposed terminal [5].

1.3.1 Hidden terminal

The hidden terminal problem is where a wireless node is within sensing range of the access point and some interfering device, but this interfering device is not sensed by (hidden from) the access point. In the diagram, where sender S2 and receiver R1 are within broadcast/transmission range; the hidden interfering terminal is sender S1. In this example, when S2 and the interfering node, S1, both transmit at the same time (as S2 and S1 cannot sense each other), receiver R2 will receive both signals at the same time, which basically becomes noise - useless, garbled data. S2 doesn’t know the message was garbled/lost, and would have to retransmit upon request. The net effect is, a TCP flow for R2 will suffer decreased throughput due to increased packet loss.
This scenario occurs due to design of the 802.11 CSMA/CA (Carrier sense multiple access with collision avoidance) protocol – if a node detects signal from other nodes, it concludes that the airspace is busy, and hence waits until a random back off time passes to retransmit. The problem here is sender S1 and sender S2 cannot detect each other; hence R2 receives packets from both senders at the same time.

### 1.3.2 Exposed terminal

![Diagram of exposed terminal problem]

An exposed terminal problem occurs when a wireless node senses that nearby nodes are transmitting at the exact same time and hence backs off from transmitting. In the diagram above, the sender denoted S1 is transmitting to a receiver denoted R1. At this time, sender S2 wants to transmit to receiver R2, but is prevented from sending because S2 can sense the signal of nearby node S1. This is due to the design of the 802.11 CSMA/CA protocol – S2 detects that S1 is sending, so it waits until a random back off time passes to retransmit. The net result is – throughput on R2 will suffer because S2 is not sending packets to R2, even though R2 can receive S2’s transmissions clearly with no interference.

### 1.4 Thesis outline

Chapter 2 describes related work in detecting wireless interference, and some solutions to this problem. Chapter 3 describes the VOIDShaper work in detail, which includes defining wireless fairness, the topology setup, its 4-stage pipeline (network capture, capture analysis,
VOID analysis, traffic shaping), and a proposed algorithm. Chapter 4 is the evaluation section, which details the test environment setup and results from five experiments.
Chapter 2: Related work

Prior work has proposed approaches to fix the two problematic scenarios, with the first step being to map the network to see wireless node interactions. Once interferers can be found, then actions can be applied to ensure fair sharing of bandwidth across nodes. But the first step isn’t easy - trying to pinpoint problematic wireless nodes usually require taking a network offline for analysis or having the clients report problems, both of which are impractical. The root problem lies in the physical location of nodes; hence the problem can only be truly solved by relocating the node to better sense neighbouring nodes. Other solutions involved designing new MAC protocols, or rate adaptation mechanisms.

2.1 Mapping out the network

A common approach is to first map out the 802.11 wireless network to find all devices and the potentially interfering ones, such as systems like Jigsaw [12,13], Micro-probing [14], PIE [15] and VOID [2]. In general, it is hard to get an instant, complete map of network nodes and activity because of its dynamic and decentralized nature. At any given time, the wireless airspace is susceptible to outside interference, which is uncontrollable and does not get factored into the network map.

Jigsaw [12] is a system which constructs a global view of network activity in a wireless network. It uses wireless sensors placed close to production access points to detect and capture network traffic. The system then analyses the network capture trace to find interference between wireless nodes. The drawback with this approach includes 1) wireless sensors need to be deployed next to all production access points which could be prohibitive due to cost, 2) algorithms are needed for clock synchronization which suggest this approach is more suitable for offline (not real-time) analysis.

In the Online Estimation of RF Interference work [14], the authors used micro-probing techniques to construct a network map. Coordinated through a central controller, special requests are sent to the access points to conduct an active test lasting 1-2ms. During this time, the network has to be silenced by sending CTS-to-self packets for the test duration.
Non 802.11 devices using the same airspace will be non compliant, hence it is hard to get absolute network silence. Another requirement is the clock synchronization between the controller and access points. A further requirement is the modification of access point software, which increases the implementation complexity.

VOID (Wireless Online Interference Detection) [2] uses a statistical method called multiple linear regression to map the interfering wireless nodes by analyzing network trace files collected at the central router which all wireless access points connect to. It is passive like Jigsaw in that it only listens to network traffic (compared to micro-probing which actively injects network packets). Hence, instead of directly measuring RF interference physically like the other approaches, it infers interference relationships by examining the subtle changes in TCP traffic in a higher layer.

By looking through the network trace, VOID can find correlations between node throughputs which could be caused by interference. If moments of increased throughput in one node repeatedly cause drops in throughput for another node, then there is potential interference between this node pair.

VOID is able to produce an interference map in real-time and only requires a few seconds to identify interferers to a victim node in one measurement round. It has been shown to correctly and effectively identify interfering devices in testbed and live networks. I will be using this approach to find a list of wireless interferers.

### 2.2 Solutions to wireless fairness

With the network map and interferers found, corrective action can be applied to alleviate interference. Techniques include traffic shaping at a central router, or timing the Access Points to send packets at a precise manner.

In the CENTAUR [16] system, which uses microprobing [14], a router/central device is used to categorize and control all traffic to be sent to the access points. This synchronization between the router and access points allows data to be sent in a precise, timely manner,
which effectively reduces interference by controlling when packets are sent (such that senders don’t transmit at the exact same time and cause packet loss). A downside to this approach is the complex changes required in the access points in order to synchronize with the router, as well as the need of microprobing for active network packet injections.

In Kan Cai's shaper paper [5, 17], a central router is used upstream to shape TCP traffic being sent to the access points. The idea is to ensure that no TCP flow is dominating the wireless airspace by introducing a fair queuing mechanism. With the network map and interferes detected, the TCP flows can be subjected to the networking fair queue, to ensure that the bandwidth is being utilized fairly. This leads to decreased utilization as a downside, but the idea is to reduce the airspace congestion such that weaker flows can achieve better throughput.

2.3 Changing the 802.11 protocol

Other approaches which require replacing a part of the 802.11 protocol (which means cooperation is needed from access points and wireless clients) include using time-based fairness [11, 18], replacing 802.11's DCF MAC protocol [19, 20], new packet fair queuing schemes [21, 22], and modifying the rate fallback/adaptation algorithm [23, 24].

Time based fairness is an idea involving changing the MAC protocol to ensure 802.11 fairness. The authors argue that the existing, popular 802.11 MAC protocol, DCF (Distributed Coordinated Function), only ensures throughput fairness when similar sized packets and similar channel conditions are seen. With the TBR (Time-based Regulator) algorithm proposed which runs on the access point, packets are scheduled such that nodes get their fair portion of the channel access time, regardless of their transfer rate. However, they do not mention whether it can treat the hidden or exposed terminal problem.
Chapter 3: The VOIDShaper engine

How should the wireless unfairness problem with traffic of different transmission rates be solved? The goal is to develop a network administration tool that adaptively recognizes problematic interference, groups interfering nodes and shapes the wireless traffic at the router to restore wireless fairness. Finding interference is handled by VOID, and a mechanism providing traffic shaping is Shaper. The hard parts are: how to group these interfering nodes, how to define wireless fairness, how to handle traffic at different txrates, what traffic shaping policies should be applied, and whether the fairness can be established.

The solution is the VOIDShaper engine - a combination of the VOID (Wireless Online Interference Detection) tool, used to detect interfering nodes, and Shaper, a traffic shaping mechanism. It will process a raw network trace file containing TCP/IP packets, format the TCP throughput data, use VOID to find a list of interfering node pairs, and apply shaping rules to the TCP traffic. Plus, it does not require modifications to clients and access points, and keeps the router implementation lightweight.

The decision to shape traffic or not will depend on factors such as: correlation coefficients from VOID, each TCP flow’s throughput, the aggregate throughput, and the cost-benefit of shaping. Wireless nodes transmitting at symmetric and different 802.11 txrates are considered.

3.1 Assumptions

This approach assumes that all the wireless access points connect through the central router, such that all traffic can be controlled. This makes Shaper possible to administer the network centrally. Only traffic going to/from the router is controlled, which means that any traffic on the wireless subnet will not be seen at the router to be shaped. And, only TCP traffic is analyzed, as it is the dominant type traffic on the Internet.

For VOID analysis to function, the network should be congested (in other words, the airspace completely saturated) such that interference will take place. This is reasonable because
otherwise the senders do not compete, and TCP flows aren’t affected and won’t show any interference patterns. Note that a flow which has been starved with little or no throughput will not be seen or analyzed by the VOID tool; however, in this case the wireless client should know to relocate to acquire a better wireless connection with the access point if he/she gets no throughput.

3.2 Defining wireless fairness

Wireless fairness can be defined as nodes getting the same throughput under similar channel conditions. However, nodes are located at different lengths from the access point, have different transmitting power levels, are subject to outside interference, and they can be sending at different transmission rates. So, in general, if we approach it at the router level (at the transport layer, which is above the 802.11 MAC protocol), we can think about fairness as not having one flow using up all the bandwidth (disproportionally) while other flows are active, even if the hidden or exposed terminal problem exists.

If two nodes are sending at the same txrate, then fairness is simply TCP flows achieving the same throughput. In the other case, at different txrates, fairness can be considered as not one flow dominating the entire airspace unfairly - if it slows down, do the other weaker nodes benefit? Conversely, all the normal nodes connected to the same access point should not have to suffer decreased, shaped throughput just because one weak node is sending at a very low rate (such as 1Mbps). In this case, the weak node should relocate and send at a higher rate - otherwise it is unfair to have all other nodes slow down to 1Mbps due to a set rate limit by shaping.

3.3 Shaping rate/fairness determination

When interference is detected, VOIDShaper does not know the sending rate of the wireless nodes. The only information available is the current TCP throughput, VOID correlation output, and the same information from previous runs of the pipeline. So, what should the shaping rate be? And how should fairness be determined considering the overall network utilization?
3.3.1 Two wireless nodes sending at the same txrate

When two nodes are sending packets at the same txrate, the amount of time it takes for the packet to be transmitted are the same. If both nodes are sending at the same txrate, for example, at 36Mbps, then each should get half the total throughput in order to achieve fairness. So, if Node A achieves 20Mbps throughput, and Node B achieves 4Mbps throughput, then after shaping, each node should receive \((20+4)/2=12\)Mbps to be fair. This is the trivial case.

However, it isn't always the case that the weaker flow can use up 12Mbps as discussed in the previous section; hence, the shaping rate should be reduced such that both nodes can get a fair throughput. Note that a naive approach to achieve fairness is to shape to the throughput of the slowest flow. For example, in this case we can just shape to 8Mbps such that each will get 4Mbps (since we know the weak flow can achieve at least this much); however, the total throughput will be 8Mbps, which makes the network severely underutilized.

3.3.2 Two wireless nodes sending at different txrates

Here, VOIDShaper has no idea these nodes are sending at different txrates. An issue here is that a wireless packet sent at a lower txrate takes much longer time than a packet of the exact same size sent at a higher txrate. So we should strive for time fairness (regardless of txrate) where nodes get the same amount of time in transmitting. If we reuse the same example as above, suppose Node A (sending at 36Mbps) achieves 20Mbps throughput, and Node B (sending at 6Mbps) achieves 4Mbps throughput. For example, if we shape both flows together to 20Mbps total, Node B will not see an increase in its throughput. Node B is likely to stay around the same 4Mbps while Node A will get 16Mbps. So, here, VOIDShaper should see that shaping did not improve Node B's throughput, and going further would reduce overall utilization without improving fairness.

This is real world example where a weaker node is sitting further from the AP, sending at 5.5Mbps, while the stronger node is closer to the AP sending at 36Mbps. Given they are interfering, should we shape to improve their fairness? In the worst case, we can end up
shaping to accommodate to the slowest flow (the lowest common denominator). Hence, at any time after shaping, if the weaker flow's throughput isn't observed to be increasing, then we should stop shaping to preserve network utilization. Otherwise, the strong flow will suffer from unnecessarily lower throughput just because a weaker node's txrate is lower.

3.4 Topology/environment
Our topology involves a hidden terminal topology, with a total of 6 nodes. As seen in the following figure, there are four wireless nodes in total, and two wired nodes. The central router is connected in a 100Mbps wired LAN to the two access points (*nodew1* and *nodew3*). The wireless clients are *nodew2* (connected to the *nodew1* access point), and *nodew4* (connected to the *nodew3* access point).

![Figure 3.1 Hidden terminal topology setup in emulab](image)

The *client* node acts as the outside Internet server which will establish a TCP stream with the wireless clients *nodew2* and *nodew4*. The 802.11 interfaces on the 4 wireless nodes run on the 802.11g protocol, with a fixed transmission rate set. The maximum useable rate is
36Mbps and is set accordingly in the experiments. Note that RTS/CTS is turned off by default as it is optional in 802.11, as well it has be shown to decrease 802.11 performance [25].

The TCP stream will send at a constant date rate of 30Mbps to each of the wireless clients, thus keeping the wireless airspace completely utilized and congested. Refer to Appendix A for the tool used to send the long-lived TCP flow.

The environment in which the experiment is conducted in is the University of Utah’s Emulab Network Testbed [26]. The figure below shows the 4th floor wireless node layout in the Merrill Engineering Building at the university; blue dots represent nodes which were in use at the time, green dots represent free, unused nodes, and red dots represent dead nodes.

![Emulab machine location map](image)

Figure 3.2  Emulab machine location map

The wireless nodes in the hidden terminal topology are mapped to physical wireless nodes when the topology becomes active. In this case, nodew1 maps to pcwf14, nodew2 maps to
pcwf12, nodew3 maps to pcwf7, and nodew4 maps to pcwf8. The client and router nodes are not shown here, but they are located in a different area of the same building.

3.5 VOIDShaper details
On the emulab router machine, we start the VOIDShaper Engine by launching a Python script which is responsible for running the entire “pipeline”. The pipeline consists of a few stages, including 1) network capture, 2) capture analysis, 3) VOID interference detection, 4) Traffic Shaping Engine. These 4 stages make up one run of the pipeline. Our proposed adaptive approach runs this pipeline repeatedly, so we can observe whether the traffic shaping done was successful or not in later runs.

![Pipeline Design](image)

**Figure 3.3** The components of the pipeline design

3.5.1 Network capture
For collecting network traces, we use the popular wireshark command line tool. The capture is started on the router node, where data from the client to the wireless node (and vice versa) passes through. The first 100 bytes of the Ethernet frame is saved, and each run lasts for 30 seconds. Traffic is generated with the DITG tool on the client, where a constant
30Mbps TCP flow is sent to each of the wireless nodes nodew2 and nodew4. That means, traffic is sent from the client to the router, and from the router to the wireless access points (nodew1, nodew3), and then lastly to the wireless devices (nodew2, nodew4).

3.5.2 Capture analysis
With the raw network trace file, we feed it into the wypy engine [29] for analysis. Wypy will take packets in the trace file, determine the packet type, and count how many packets are being sent in a given time window - the duration used is 50 milliseconds. Knowing the amount of packets sent, we can determine the instantaneous bandwidth of a certain TCP flow across time windows. The output is a time bucket file containing the throughput information of all TCP flows which are observed.

3.5.3 VOID analysis
With the throughput information of each TCP flow, VOID uses a statistical method called multiple linear regression (MLR) to analyze all flows against each other, in order to determine whether one flow is interfering with multiple other flows. The strongest correlation occurs whenever one node is sending with increased throughput, and throughput drops for another node at the exact instant.

3.5.3.1 Throughput function and the coefficients
In VOID, the interference relationships between pairs of nodes are assumed to be linear. As such, for a particular victim node, \( y \), the throughput relationship can be expressed with Equation 1 [2], where:

\[
y = x_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k
\]  

(1)

\( x_0 \) is the throughput victim node \( y \) would achieve if all other interferers aren't active,
\( x_i \) is the throughput of interfering node \( i \),
\( k \) represents the number of interferers,
\( \beta_i \) is the coefficient representing the impact of interferer \( i \) on the victim's throughput.
If a node \( i \) is interfering with the victim node, then \( \beta_i \) should be a negative number, suggesting that an increase of this interferer node's throughput will be reflected by a decrease in the victim node's throughput. If the coefficient is a positive number, then interferer node \( i \) is possibly helping the victim node.

### 3.5.3.2 Sample VOID xml output

The output of VOID is an interference map (xml file) of the nodes containing the correlation values between each pair of TCP flows. Below is an example output.

```xml
<graph edgedefault="undirected">
  <!-- data scheme -->
  <key attr.type="string" id="ip" for="node" attr.name="ip"/>
  <key attr.type="double" id="MC_Value" for="edge" attr.name="MC_Value"/>
  <!-- nodes -->
  <node id="1">
    <data key="ip">10.1.1.3</data>
  </node>
  <node id="2">
    <data key="ip">10.1.4.3</data>
  </node>
  <!-- edges -->
  <edge source="10.1.4.3" target="10.1.1.3">
    <data key="MC_Value">-0.32</data>
  </edge>
</graph>
```

This is the file generated for the TCP Flow 10.1.1.3 from one of the runs in the two node hidden terminal topology. We can see the interferer nodes listed one by one – the other receiver node (or TCP flow) being 10.1.4.3. There is one edge, from the other node (10.1.4.3) to this one (10.1.1.3). The MC_Value, -0.32, is the coefficient \( \beta_i \) that is shown in Equation 1 earlier. It is important to understand this MC_Value which will be listed in many of the experimental results.
3.5.3.3 Visualizing the network traffic

In the figure above, we can easily visualize how many packets are sent from the router to the wireless clients. On the x-axis are sample points taken every 50 milliseconds. On the y-axis is the number of packets (throughput), with each packet being 1500 bytes.

The interesting points which show the strongest interference lies around points \( x = \{65, 108, 141, 240\} \), where the red arrows point to. Notice there is a strong flow sending at a high rate and a weak flow not being able to send any packets at that exact instant. In essence, the weak flow is starved when the strong flow is sending. This is the hidden terminal problem where the strong flow saturates the airspace while the weak node is receiving data from its access point. Anytime the strong flow is using the airspace, the weak flow will be affected since its signal is affected at the Media Access Control (MAC) layer.

Figure 3.4 Plot of network capture data showing interference
3.5.4 Traffic shaping engine (shaper)

After the list of interfering devices is found, a series of rules can then be used to decide how to shape the traffic to ensure fair bandwidth sharing. By reading the correlation value, it can be determined if one flow is affecting another. A throughput limit can be calculated based on the TCP throughput of group of nodes that are interfering.

Shaper works because the amount of packets being sent wirelessly is controlled at the router by a throughput limit. Since the router is using a well understood fair queuing scheme, only a capped amount of wired packets from TCP flows queue up at the router. The packets then leave the router for the wireless access points to be sent to receivers. With less wireless packets being pushed into the airspace, it is less likely for 802.11 MAC-level congestion to occur [5], thus alleviating the wireless unfairness problem.

3.5.4.1 Grouping the interfering nodes for shaping

The VOID analysis results will provide a list of interfering nodes. This list of interfering nodes are grouped together and shaped together. This is a simple approach taken especially with just two TCP flows. Further work can explore more TCP flows and how grouping should be handled, especially in between VOIDShaper iterations.

3.5.4.2 Shaping the TCP flows together

The shaping approach used contains a Hierarchical Token Bucket at the root, with the group’s maximum bandwidth set. Then, we have one child HTB classes under the root with the same bandwidth limit set. And, under the HTB class at the leaf, a SFQ is used to ensure fair queuing of packets across all flows. The diagram below depicts the traffic shaping structure.
Both flows will share the 20Mbps fairly through the SFQ. By shaping them together, the overall utilization is kept higher - if the weaker flow cannot use up all the allotted bandwidth due to its channel conditions or if it stops sending traffic altogether, the stronger flow can use up to the maximum allotted bandwidth (up to 20Mbps).

### 3.5.4.3 An alternative: shaping the flows individually

This alternative approach uses a Hierarchical Token Bucket at the root, with the group’s maximum bandwidth set. Then, child HTB classes are placed under the root with the limit being the node’s fair share of bandwidth. And under each HTB class, a SFQ is used where a specific flow is matched to. The diagram below depicts the traffic shaping structure.
For example, if we have 2 nodes in the hidden terminal case, with a strong flow achieving 19Mbps throughput, and a weak flow at 1Mbps, a shaping rule can be applied with each node getting a fair share of the aggregate 20Mbps bandwidth - in this case, 20Mbps / 2 = 10Mbps. The SFQ will ensure that TCP flows are getting a fair share of the allotted bandwidth. This fairness improvement, of course, can lead to lower network utilization.

Compared to shaping the flows together, the stronger flow is not limited to just 10Mbps if the weaker flow cannot use up all of its share, and the SFQ will take care of the fair scheduling of packets from both flows. The drawback, as seen in the results section, is that the strong flow can still dominate the airspace if the bandwidth limit is set incorrectly. In the results section, we observe that setting the limit to 85% of the total bandwidth is appropriate.

Lastly, since the traffic is fed through a fair queue, the VOID correlation becomes almost perfect (in the range of [0.95,1.05]) and hence masking the interference; in this alternative shaping approach, VOID would show interference has gone after shaping each flow individually. These observations are discussed in detail in the results section.

### 3.6 Proposed traffic shaping algorithm

After stage 3 of the VOIDShaper pipeline, the flow throughput information and VOID output is passed into the traffic shaping engine. If VOID did not find any correlation, then no action needs to be taken (traffic shaping does not take place). On the other hand, if interference is found, outlined below is how VOIDShaper should determine what rate to shape to.

```
Given the list of nodes that are interfering, are they SharingBandwidthFairly()?
  If YES → traffic shaping is not needed
  If NO → has this group of nodes been shaped at the previous iteration?
    If YES → check with the CostBenefitFunction() to see if it's worth shaping
      If YES → shape to new rate
      If NO → no action
    If NO → shape to 85% of TotalBandwidth()
```

Traffic Shaping Algorithm
When the algorithm first runs, given a list of nodes that are interfering, and if they are not sharing bandwidth fairly, the algorithm will shape the TCP flows. The shaping speed parameter is determined to be 85% from experimental results. On the next iteration, if the nodes aren’t interfering anymore (VOID did not correlate the TCP flows as interfering), or if the TCP throughput shows fair sharing, then no action is taken. Otherwise, the algorithm will check with the Cost Benefit Function to see if any further shaping action should be taken – taking into consideration the benefits/tradeoffs of applying stricter shaping rules. Each method is described in detail below.

**TotalBandwidth method:**
The total bandwidth is determined by adding up the throughput achieved for all the flows which are found to be interfering with each other. For example, if flow 1 is getting 20Mbps, and flow 2 is getting 5Mbps, the total bandwidth would be 20Mbps + 5Mbps or 25Mbps.

**SharingBandwidthFairly method:**
To see if the two flows are using their bandwidth fairly, we see if they are both within 10% of the average throughput. To generalize for n number of flows, each flow's throughput would be checked to see if it is within ±10% of the average throughput of all flows which is defined as follows:

\[
\text{Is } \text{throughputOfFlow}(i) \text{ within range of:} \]

\[
\left[ \frac{90}{100n} \sum_{i=1}^{n} \text{throughputOfFlow}(i) , \space \frac{110}{100n} \sum_{i=1}^{n} \text{throughputOfFlow}(i) \right]
\]

**CostBenefitFunction method:**
Looking at the previous VOIDShaper run, after the shaping rate was applied, did the weaker flow see an improvement in the TCP throughput? If there is no increase, or throughput remained the same, then there is no benefit to shape further, as the stronger flow will suffer lower throughput at a lower shaping rate.

If there is an observed throughput increase of more than 10% for the weak flow, then there is some benefit to shaping to a new rate which will be 85% of the aggregate throughput. Here
the threshold of 10% is chosen arbitrarily. That is to say, if we lose 15% of network utilization for all the interfering nodes found to get 10% increase in throughput for the weak flow, then it is worth shaping. Here, the method can decide to shape to a new rate other than 85% if there are other factors to consider, and return that rate to the VOIDShaper algorithm.

Otherwise, if the throughput increase is less than 10%, then continuing to shape to a lower rate will not help the weak flow, which is likely sending at a lower txrate than the rest of the interfering flows. In other words, it's not worth losing network utilization just because a weak flow is sending at a lower rate - as mentioned before, the strong flows should not have to slowdown just for this weak node.

3.7 When to remove the shaping rules?
Or, similarly, when to adjust the shaping rate upward? Ideally, shaping rules should be removed if interference no longer exists, if the nodes have relocated, if different nodes are found to be interfering (changing interference map), if a certain time limit has been reached, or when the TCP flows are no longer active. However, whenever a shaping rule is removed, interference can come back, and it is possible to flip-flop between shaping and not shaping.

It makes sense for the algorithm to re-examine at the current shaping rules whenever it isn't shaping the traffic to a certain rate. A more detailed study for this policy should be examined, especially for a scenario with many nodes.
Chapter 4: Evaluation

In this chapter, five experiments are described to show whether VOIDShaper can effectively detect wireless interference and examine the results after applying shaping rules. The correct action can be to shape the traffic at some maximum bandwidth limit, to shape to an even lower rate than the previous iteration, or not to shape at all. The first three experiments look at wireless senders sending at the same transmission rate (all nodes at 36Mbps, or all nodes at 11Mbps), with different shaping techniques and VOIDShaper taking different actions.

The last two experiments examine wireless senders sending at different txrates (such as 36Mbps versus 5.5Mbps). The strong flow can be set to 36Mbps, with the weak flow at 5.5Mbps, and vice versa. These are two different scenarios as a weak flow's maximum throughput will be bound to the transmission rate; hence a weak flow sending at 5.5Mbps will be interesting to observe.

Can VOIDShaper successfully detect interference with traffic at symmetric txrates? Can the Emulab environment be setup with constant interference for experiments? What will the interference output look like for traffic at different txrates? How does VOIDShaper report the correlation value - before and after shaping? What should the shaping rate be set to? Is there a case where interference is detected but no shaping is necessary? How will the shaping affect throughput with traffic of different txrates? And, finally, are the decisions made by the proposed traffic shaping algorithm effective in shaping away wireless interference?

Description of the result columns

Below is a sample table header used for the experiment results.

<table>
<thead>
<tr>
<th>Result</th>
<th>Scenario</th>
<th>Strong→Weak Correlation</th>
<th>Weak→Strong Correlation</th>
<th>Weak Flow TP (Mbps)</th>
<th>Strong Flow TP (Mbps)</th>
</tr>
</thead>
</table>

The "Result" column gives a number for referencing a specific result in discussions. Next, the "Description" column shows what shaping actions were applied for that scenario; if the
description is blank, then it is part of a series of runs in a scenario which is listed before the current run.

The "Strong→Weak Correlation" column shows if the strong flow is interfering with the weak flow. This correlation value is the "MC_Value" (discussed in Section 3.5.3.1) collected from VOID output, which is a coefficient measuring the correlation in changes in the strong flow's throughput with changes the weak flow's throughput. Conversely, the "Weak→Strong Correlation" column shows how much the weak flow is impacting the strong flow. For either column, if the correlation number is negative, that means interference is detected - meaning the other flow is affecting our throughput negatively. A value of "None" means no interference was detected.

The last two columns show the throughput (TP) of the weak flow and the strong flow achieved in Mbps. Throughput is measured by the average data rate computed from a network trace file for a particular result.

4.1 Experiment setup

All the experiments were conducted on the Emulab machines in the 2 node hidden terminal topology as described in section 3.4. There are two TCP flows where one is the strong (faster) flow, and the other is weak (slower) flow. And, the shaping method (shaping the TCP flows together) described in 3.5.4.2 will be used for all the experiments, except in the second experiment when the alternative method is explored.

The first scenario is always the beginning of the experiment and had no shaping actions applied. It contains the first 5 results, and the goal is to see that interference can be constantly detected by VOIDShaper. If interference cannot be constantly observed (correlation value shows "None"), the experiment is tried again at a later time under different network conditions. The challenges to setting up a successful environment are described in the following section.
Each trial within a scenario in the experiment was run in succession; so, for example, Result #2 was obtained immediately after Result #1. The same practice applies to scenarios - they were ran one after another without breaks. This ensures the experiment finishes running as soon as possible; otherwise, wireless network conditions change and the topology may not have the desired interfering patterns found at the beginning of the experiment.

In practice, if VOIDShaper is running in real time, then the algorithm would apply necessary actions from results in a single iteration, not from the average of 5 runs. In these experiments, each scenario is repeated five times to show if the interference is detected constantly, and whether the TCP throughputs are stable. In other words, the results tables show what would happen if VOIDShaper took an action after 5 runs, and what the outcome of those actions would be.

4.1.1 Wireless node selection
To ensure interference takes place, the first step is to carefully chose the wireless nodes in Emulab. Their physical location served as a good estimate for how far senders can reach/impact other nodes. Once the nodes are deployed, throughput tests are ran on the senders to make sure there is a "strong" sender and a "weak" sender.

A successful topology will have a weak sender achieving 20-23Mbps throughput by itself and a strong sender achieving 22-24Mbps no matter if the weak sender is active. However, once the strong sender is active, the weak sender will only be able to achieve 3-5Mbps, since the affected receiver is being impacted by both the packets coming from the strong sender and the weak sender. Most importantly, VOIDShaper should detect interference in this case, and the experiment can continue. In the wireless traffic at different txrates, the same process takes place - there should be a weak sender which is affected by the presence of a strong sender, and VOIDShaper needs to detect interference.

4.1.2 Emulab challenges
It's not trivial to get the exact environment with the right conditions in Emulab. The first challenge is the availability of nodes; in times of heavy usage, the hidden terminal topology
is impossible to deploy because it could be short just one of the handpicked nodes. Using other nodes will change dynamics of the topology. Given the nodes were available, sometimes the hidden terminal property isn't satisfied - both senders may be strong, and obviously no interference will be found. It takes a lot of tweaking to make it useable, such as changing the txrates, lowering the strong sender's antenna power, or picking new wireless nodes altogether.

It should be noted that the wireless testbed is sharing the same airspace as the university's production wireless network. The "iwlist" tool can be used to get detailed information from the wireless network interface, such as the list of networks that can be detected. Running the "iwlist" command on the receiver nodes shows at least 10 other networks (SSIDs) detected. Hence, a topology deemed hidden terminal this hour might lose this property in the next. Ensuring that the topology has the desired interference properties before running experiments is very important.

When dealing with traffic sent at different txrates, the topology that was setup for symmetric txrates cannot be used. At different txrates, the interference range of a sender changes – at lower txrates, the packets are more resilient to noise from other senders. To get interference to take place, the transmission power or txrates have to be adjusted, even the wireless nodes have to be handpicked again at times. This is why the scenarios were ran for 5 trials to weed out potential outside interference beyond our control – if an unusual test result is observed, it is thrown out and not used to calculate averages.

4.2 Interfering traffic at symmetric transmission rates (36Mbps)
The first experiment shows VOIDShaper working in the most intuitive case in the hidden terminal topology – two senders sending at the same transmission rate of 36Mbps. Can VOIDShaper find the interference constantly across five runs? And what is observed after shaping is applied?
VOIDShaper is expected to find the interfering nodes, by showing negative correlation in the results. And, if interference is found, a shaping rate is applied to shape both TCP flows. It is expected that both flows will share bandwidth fairly after shaping rules are applied.

The experiment begins with both wireless senders transmitting traffic at 36Mbps, with no shaping involved at the router. Note that the weak TCP flow achieves a throughput of 23Mbps without the strong flow running, and the strong flow achieves 25Mbps by itself. Each scenario was run for 5 times (except for the last scenario, where only 4 trials are recorded).

<table>
<thead>
<tr>
<th>Result</th>
<th>Scenario</th>
<th>Strong→Weak Correlation</th>
<th>Weak→Strong Correlation</th>
<th>Weak Flow TP (Mbps)</th>
<th>Strong Flow TP (Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No shaping</td>
<td>-0.25</td>
<td>-0.77</td>
<td>5.447</td>
<td>20.047</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>-0.11</td>
<td>-0.28</td>
<td>4.304</td>
<td>21.395</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>-0.46</td>
<td>-0.83</td>
<td>3.973</td>
<td>21.755</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>-0.09</td>
<td>-0.44</td>
<td>4.202</td>
<td>21.082</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>-0.16</td>
<td>-0.55</td>
<td>2.994</td>
<td>20.857</td>
</tr>
<tr>
<td>6</td>
<td>Shape to 21Mbps total</td>
<td>None</td>
<td>None</td>
<td>10.519</td>
<td>10.528</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>None</td>
<td>None</td>
<td>10.303</td>
<td>10.322</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>-0.93</td>
<td>-0.95</td>
<td>10.235</td>
<td>10.370</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>-0.62</td>
<td>-0.64</td>
<td>10.305</td>
<td>10.322</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>-0.78</td>
<td>-0.68</td>
<td>10.315</td>
<td>10.295</td>
</tr>
</tbody>
</table>

Table 4.1  Traffic shaping and VOID analysis results for shaping flows together

When the experiments begin (result #1-5), the weak flow is observed to be getting only 3-4Mbps throughput in the presence of the strong flow. As for the strong flow, it's actually giving up about 4Mbps with the weak flow present. In all results #1-5, VOID is able to detect interference, with negative correlation values. This shows interference is taking place, and the topology can continue with the next step.

Since VOIDShaper detects interference, it will decide to shape both flows together to 85% of the aggregate bandwidth. The shaping rate is calculated to be 21Mbps total; for example, in result #4, 85% of the aggregate bandwidth is \((4.202+21.082)/2 \times 0.85\) or rounded to 21Mbps.
The results are interesting when shaping to 21Mbps - the interference is not detected by VOID at times. This is the side effect of shaping both flows together through the same SFQ - the packets are scheduled fairly and it makes sense that if one flow gets more packets into the queue, the other flow will suffer (since it lost to the other flow the same amount of packets). The important point here is the flows are sharing completely fairly, each using up 50% of the allotted total shaping rate. At that point, VOIDShaper stops shaping, even if interference is detected.

Figure 4.1  The effect of shaping rate on throughput of two TCP flows

In the above figure, the throughput of the two TCP flows and their aggregate throughput are plotted against different shaping rates. At the beginning, when there is no shaping, the strong flow (achieving 21.0Mbps) clearly dominates the weak flow (at 4.2Mbps).
When traffic shaping is applied, for example at 26Mbps and 25Mbps, the aggregate throughput remains high above 24Mbps, with the strong and weak flow's throughput gap smaller. At 24Mbps and 23Mbps, the aggregate throughput decrease, and the strong/weak flow throughput gap becomes even smaller.

At 22Mbps, both the strong and weak flow starts to share the airspace completely fairly at 10.7Mbps. From a previous study [5], approximately 85% of the total bandwidth is a suitable shaping rate for bandwidth use equality. In this case, 85% of 24.1Mbps is 20.5Mbps, which is the bandwidth limit that should be set by our shaping policy.

The following graph is a plot of the throughput of both TCP flows during the first run in Table 4.1., where 10.1.4.3 is the strong flow, and 10.1.1.3 is the weak flow, both initially sending at 36Mbps, with no shaping. The x axis represents time, and in this case, throughput sample points from time [50,250] are used. Notice the moments in time where the strong flow gets an sudden burst of speed, and the weak flow decreases at that exact time. Conversely, the weak flow has the same effect on the strong flow - they are both fighting for the available airspace, hence interference is detected.
The following graph is a plot of the throughput of both TCP flows during the 8th result in Table 4.1, where 10.1.4.3 is the strong flow, and 10.1.1.3 is the weak flow, both sending at 36Mbps still, with shaping set to 21Mbps total. The strong flow's effect on the weak flow (the VOID correlation MC_value) is $-0.93$, and the weak flow's effect on the strong flow is $-0.95$. The graph shows the throughput sample series from time 50 to 250.
The figure above shows a flat throughput series plot since both flows are being shaped at the router at 21Mbps total. At one point, around time 216, the strong flow gets a sudden surge in speed with the weak flow having the exact same decrease. All the minute fluctuations in both flows are subject to the TCP traffic being shaping together - hence the strong correlation in these graphs, with MC_Values of -0.93 one flow and -0.95 for the other.

4.3 Alternative shaping method: shaping each flow individually
This experiment involves shaping each flow individually - this alternative shaping method is described in section 3.5.4.3. This experiment shows the effect on the correlation output values produced by VOIDShaper under this different shaping method. What does the correlation output look like, and, is it a good shaping method to use?
The first 5 results show the VOID output and throughput of the two TCP flows at a symmetric txrate of 36Mbps. In the second scenario (Result #6-10), the TCP flows are shaped to 14Mbps individually at the router. Both scenarios are repeated 5 times to show that the interference is consistent, and to weed out potential interference beyond our control. The weak flow can achieve about 21Mbps by itself, and the strong flow can achieve 25Mbps by itself.

<table>
<thead>
<tr>
<th>Result</th>
<th>Scenario</th>
<th>Strong→Weak Correlation</th>
<th>Weak→Strong Correlation</th>
<th>Weak Flow TP (Mbps)</th>
<th>Strong Flow TP (Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No shaping</td>
<td>-0.06</td>
<td>-0.11</td>
<td>3.401</td>
<td>25.292</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>-0.08</td>
<td>-0.22</td>
<td>2.999</td>
<td>24.497</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>-0.10</td>
<td>-0.31</td>
<td>3.017</td>
<td>24.520</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>-0.10</td>
<td>-0.24</td>
<td>3.127</td>
<td>24.901</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>-0.07</td>
<td>-0.18</td>
<td>3.274</td>
<td>24.929</td>
</tr>
<tr>
<td>6</td>
<td>Shape each flow to 14Mbps</td>
<td>None</td>
<td>None</td>
<td>8.943</td>
<td>14.032</td>
</tr>
<tr>
<td>7</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>9.104</td>
<td>14.032</td>
</tr>
<tr>
<td>8</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>8.861</td>
<td>14.032</td>
</tr>
<tr>
<td>9</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>8.352</td>
<td>13.732</td>
</tr>
<tr>
<td>10</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>8.444</td>
<td>13.703</td>
</tr>
</tbody>
</table>

Table 4.2 Traffic shaping and VOID analysis results for shaping flows individually

In the first scenario, Result #1-5, the weak flow is clearly affected by the presence of the strong flow, as the throughput is reduced to about 3Mbps. VOID output in the 2nd and 3rd column shows that the two flows are interfering with each other with a negative correlation value, similar to the previous experiment when no shaping is done.

In the second scenario, the shaping rate is determined by the total bandwidth of the 2 TCP flows, divided by 2, which is (25Mbps+3Mbps)/2 or 14Mbps. We expect each TCP flow to use up the 14Mbps, which would be perfectly fair. Upon activating the traffic control/shaping rules at the router and allocating 14Mbps for each flow, the immediate result seen is that interference goes away completely, which is different from shaping the two TCP flows together.

The throughput of the strong flow is restricted to 14Mbps, and immediately the weak flow can sustain higher throughput, close to 9Mbps. This is because the strong flow is no longer
dominating the airspace unfairly; as such, the weak flow's receiver is able to receive more packets without all the noise from the strong flow.

The two flows still are not sharing fairly; the strong still has an unfair advantage. However, the fact that the weak flow can only achieve 21Mbps alone is considered, that is 4Mbps slower than the strong flow. The 4Mbps loss is likely due to the physical distance separation of the 2 wireless nodes, network channel conditions at the time of experiment, or outside interference. As such, if the loss of 4Mbps is factored in, the weak flow is actually using up 9Mbps + 4Mbps or 13Mbps of its share. Note that there is no need to shape to a higher limit, as both flows can only achieve 28Mbps total. And, shaping to a lower speed would decrease the overall utilization.

Compared to the other shaping method (shaping TCP flows together), this method is more interesting in that correlation is shown to be absent (None) immediately after shaping. This can be a signal that interference has been alleviated. However, the overall network utilization is kept low, as the strong flow cannot use up the slack that the weak flow has left - leaving the airspace not congested. And, if the strong flow were to stop transmitting at anytime, the weak flow can only send at a maximum of 14Mbps until the shaping rules have been removed - again lowering utilization.

### 4.4 Interfering traffic at symmetric transmission rates (11Mbps), fair sharing

Here is a case where the wireless nodes are sending at the same txrate (11Mbps) but considered sharing the bandwidth fairly. VOIDShaper is expected to detect the interference and conclude from the throughput measurements that no shaping action is needed.

<table>
<thead>
<tr>
<th>Result</th>
<th>Scenario</th>
<th>Strong→Weak Correlation</th>
<th>Weak→Strong Correlation</th>
<th>Weak Flow TP (Mbps)</th>
<th>Strong Flow TP (Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No shaping</td>
<td>-0.98</td>
<td>-0.12</td>
<td>6.429</td>
<td>7.156</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>-1.93</td>
<td>-0.13</td>
<td>6.524</td>
<td>7.168</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>-0.54</td>
<td>-0.04</td>
<td>6.522</td>
<td>7.123</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>-0.84</td>
<td>-0.07</td>
<td>6.413</td>
<td>7.179</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>-1.30</td>
<td>-0.10</td>
<td>6.452</td>
<td>7.187</td>
</tr>
</tbody>
</table>

*Table 4.3 Throughputs and VOID analysis results for 11Mbps wireless traffic*
In all 5 results, where no shaping takes place, interference is constantly detected by VOIDShaper. The negative correlation numbers are strong at times, such as during result #2 and #5 for the weak flow. Looking at the throughputs, they are all within 10% of the average throughput. This means the SharingBandwidthFairly() method described in section 3.6 would decide that this is fair sharing and hence, no actions are needed.

The experiment concludes and shows that even if interference is detected, the throughputs can show the flows are sharing in a fair way and no action is needed. The VOIDShaper shaping algorithm is able to take care of this scenario, by deciding if bandwidth usage is fair in face of interference.

4.5 Interfering traffic at different txrates, with strong sender at slower rate (5.5Mbps) and weak sender at faster rate (36Mbps)

In this experiment, two nodes are sending traffic at asymmetric rates – the strong sender is sending at 5.5Mbps, and the weak sender is at 36Mbps. What correlation values will be detected in this case? And can shaping make the senders in this scenario share bandwidth fairly?
From result 1 to 5, where there is no shaping, the strong flow is dominating with its 5.5Mbps txrate; the weak flow is only able to achieve roughly 2Mbps. When VOIDShaper sees this interference, it will see that they aren't sharing bandwidth equally, and hence it will shape to 85% of the total bandwidth, which is approximately \((2+4)*0.85\) or 5Mbps.

After the shaping action is applied, in result 6 to 10, the two flows are sharing perfectly fairly at the TCP level. Their throughputs are almost identical at 2.5Mbps. VOIDShaper, not knowing their txrates were asymmetrical, has successfully fixed the unfairness which victimized the weaker node, which was sending at a higher rate than the stronger node. This scenario behaves very much like the previous experiments when the wireless traffic was sent at symmetric rates.

VOIDShaper was able to detect the interference, showing strong negative correlation values. The shaping decision to 5Mbps total in this experiment allowed both senders to share the bandwidth equally.
4.6 Interfering traffic at different txrates, with strong sender at faster rate (36Mbps) and weak sender at slower rate (5.5Mbps)

In this experiment, again two nodes are sending at asymmetric rates. The strong flow is using a txrate of 36Mbps, while the weak flow's txrate is 5.5Mbps. As always, 5 trials were conducted for each scenario. For reference, the weak flow can achieve 3.92Mbps by itself, and the strong flow can achieve 24Mbps on its own.

This is the more complicated experiment, as it was hard to get constant interference from Emulab. It took weeks, even months, to set up the right environment. Can VOIDShaper detect interference in this case, before and after shaping? And, can the fairness be improved with the weak sender transmitting at such a low rate (5.5Mbps)? What happens if shaping is done to the speed of the slowest TCP flow?
<table>
<thead>
<tr>
<th>Result</th>
<th>Scenario</th>
<th>Strong→ Weak Correlation</th>
<th>Weak→ Strong Correlation</th>
<th>Weak Flow TP (Mbps)</th>
<th>Strong Flow TP (Mbps)</th>
<th>Aggregate TP (Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>36Mbps VS 5.5Mbps</td>
<td>-0.03</td>
<td>-0.50</td>
<td>3.598</td>
<td>21.725</td>
<td>25.325</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>-0.04</td>
<td>-0.45</td>
<td>3.662</td>
<td>22.783</td>
<td>26.445</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>-0.05</td>
<td>-0.60</td>
<td>3.534</td>
<td>21.807</td>
<td>25.340</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>-0.03</td>
<td>-0.30</td>
<td>3.638</td>
<td>23.063</td>
<td>26.701</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>-0.07</td>
<td>-0.56</td>
<td>3.536</td>
<td>22.672</td>
<td>26.208</td>
</tr>
<tr>
<td>5a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Averages:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Shape to 22Mbps</td>
<td>-0.97</td>
<td>-0.96</td>
<td>3.796</td>
<td>17.791</td>
<td>21.587</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>-0.95</td>
<td>-0.99</td>
<td>3.871</td>
<td>18.198</td>
<td>22.069</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>-0.94</td>
<td>-0.97</td>
<td>3.913</td>
<td>18.057</td>
<td>21.969</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>-0.91</td>
<td>-0.99</td>
<td>3.828</td>
<td>17.694</td>
<td>21.522</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>-0.94</td>
<td>-0.98</td>
<td>3.861</td>
<td>18.208</td>
<td>22.069</td>
</tr>
<tr>
<td>10a</td>
<td></td>
<td>Averages:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10b</td>
<td></td>
<td>Gain/Loss from previous:</td>
<td></td>
<td>0.260</td>
<td>-4.420</td>
<td>-4.160</td>
</tr>
<tr>
<td>11</td>
<td>Shape to 18Mbps</td>
<td>-0.96</td>
<td>-1.02</td>
<td>3.896</td>
<td>14.136</td>
<td>18.033</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>-0.97</td>
<td>-1.01</td>
<td>3.908</td>
<td>13.829</td>
<td>17.737</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>-0.98</td>
<td>-0.99</td>
<td>3.961</td>
<td>14.071</td>
<td>18.032</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>-1.00</td>
<td>-0.98</td>
<td>3.954</td>
<td>14.078</td>
<td>18.032</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>-0.98</td>
<td>-1.00</td>
<td>3.884</td>
<td>13.850</td>
<td>17.734</td>
</tr>
<tr>
<td>15a</td>
<td></td>
<td>Averages:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15b</td>
<td></td>
<td>Gain/Loss from previous:</td>
<td></td>
<td>0.067</td>
<td>-3.997</td>
<td>-3.927</td>
</tr>
<tr>
<td>16</td>
<td>Shape to 15Mbps</td>
<td>-0.99</td>
<td>-1.00</td>
<td>4.040</td>
<td>10.991</td>
<td>15.031</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>-0.96</td>
<td>-1.03</td>
<td>3.962</td>
<td>11.068</td>
<td>15.030</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>-0.93</td>
<td>-1.05</td>
<td>3.942</td>
<td>10.878</td>
<td>14.820</td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>-0.97</td>
<td>-1.01</td>
<td>4.022</td>
<td>10.919</td>
<td>14.941</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>-1.00</td>
<td>-0.99</td>
<td>3.940</td>
<td>10.880</td>
<td>14.819</td>
</tr>
<tr>
<td>20a</td>
<td></td>
<td>Averages:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20b</td>
<td></td>
<td>Gain/Loss from previous:</td>
<td></td>
<td>0.060</td>
<td>-3.046</td>
<td>-2.985</td>
</tr>
</tbody>
</table>

Table continued on next page
In the table above, for the first scenario, Results #1-5, the weak flow achieves a not-so-weak 3.58Mbps on average and the strong achieves 22.4Mbps. The weak flow can sustain such a high throughput because it is sending at 5.5Mbps rather than 36Mbps - a low 5.5Mbps is much more resilient to the strong node's interference. So, the presence of the strong flow affects the weak TCP flow, slowing it down by only 340Kbps. Constant interference is detected, with the weak flow having a stronger impact on the strong flow than the other way around.

In the first part of the table (Result #1-20), average values (in Result #5a,10a,15a,20a) were added to each scenario to aid the analysis for this experiment. In the Result #10b,15b,20b, the "Gain/Loss from previous" is calculated by taking the difference of the average from that scenario with the average from the previous scenario. It is easy to see the trend that shaping further than 22Mbps showed very tiny speed improvement for the weak flow, with huge decreases in aggregate throughput.

After the initial scenario, the next scenarios tested were: shaping both flows to a total of 22Mbps, 18Mbps, 15Mbps, 8Mbps, and 4Mbps. 22Mbps is 85% of the total of the initial run, which is what VOIDShaper would have shaped to. 18Mbps is the next step (85% of

<table>
<thead>
<tr>
<th>Result</th>
<th>Scenario</th>
<th>Strong→Weak Correlation</th>
<th>Weak→Strong Correlation</th>
<th>Weak Flow TP (Mbps)</th>
<th>Strong Flow TP (Mbps)</th>
<th>TP Delta (Kbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>Shape to 8Mbps</td>
<td>None</td>
<td>None</td>
<td>4.009</td>
<td>4.005</td>
<td>4.329</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>-1.01</td>
<td>-0.96</td>
<td>3.996</td>
<td>3.977</td>
<td>19.668</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>-1.03</td>
<td>-0.94</td>
<td>3.957</td>
<td>4.014</td>
<td>56.726</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>-1.00</td>
<td>-0.87</td>
<td>3.980</td>
<td>3.989</td>
<td>8.303</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td>-0.91</td>
<td>-1.03</td>
<td>3.983</td>
<td>3.983</td>
<td>0.363</td>
</tr>
<tr>
<td>26</td>
<td>Shape to 4Mbps</td>
<td>-0.97</td>
<td>-1.01</td>
<td>1.999</td>
<td>2.007</td>
<td>7.339</td>
</tr>
<tr>
<td>27</td>
<td></td>
<td>None</td>
<td>None</td>
<td>2.005</td>
<td>2.002</td>
<td>2.857</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td>-0.98</td>
<td>-1.01</td>
<td>2.003</td>
<td>2.003</td>
<td>0.708</td>
</tr>
<tr>
<td>29</td>
<td></td>
<td>-0.98</td>
<td>-1.00</td>
<td>1.997</td>
<td>2.009</td>
<td>12.363</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>-0.96</td>
<td>-1.03</td>
<td>1.992</td>
<td>2.014</td>
<td>21.533</td>
</tr>
</tbody>
</table>

Table 4.5 Throughputs and VOID analysis results for wireless traffic at different txrates
22Mbps), and 15Mbps was 85% of 18Mbps, to show what would happen had VOIDShaper continued shaping down.

After shaping once at 22Mbps, the weak flow only gained 273Kbps on average. The strong flow lost 4.4Mbps, reducing the network utilization by 4.16Mbps. In this "Shape to 22Mbps" scenario (in any of the five runs), the proposed VOIDShaper algorithm would not apply any more shaping for the next step because of the cost-benefit decision. By looking at the previous iteration in VOIDShaper, it would have determined that the weak flow only achieved a marginal 273Kbps more (on average in our test runs), whereas the strong flow had to suffer a decrease of 4Mbps. This increase of 273Kbps is less than an 10% of the weak flow's throughput, hence VOIDShaper will decide not to shape it any further.

To observe what would happen if VOIDShaper kept shaping, the experiment continued to shape down to 18Mbps and 15Mbps to see if there is anything worth noting. The weak flow's average throughput did increase - albeit a negligible 60Kbps. The strong flow continues to experience a lower throughput as it is shaped together with a weak flow stuck at a lower txrate. The aggregate throughput decreases as expected.

Another shaping rate to experiment is at 8Mbps - to study the effects of naive shaping, where the rate is set to the throughput of the lowest node multiplied to the number of flows. In this case, it would be 3.6Mbps * 2 which is rounded up to 8Mbps. Both flows share perfectly fairly here, with no notions of a "strong" or "weak" flow. VOIDShaper would decide not shape in this scenario.

When continuing to shape down to the throughput of the slowest node (3.6Mbps rounded to 4Mbps), the flows continued to share perfectly fairly. One observation about this scenario (shaping to 4Mbps total) and the previous (shaping to 8Mbps total) is that out of the 5 trials, there is one case where VOIDShaper did not detect interference. This again could be due to the outside wireless interference. In any case, they are sharing fairly, so there is no need to shape regardless of the VOID output.
This is the most extensive experiment where multiple shaping rates were tested. By observing the input to VOIDShaper, and running through its algorithm, its next step can be determined. And by implementing those scenarios (as if VOIDShaper took that step), we can see the VOID output and the TCP flow throughput outcome.

In this experiment, VOIDShaper is shown to constantly detect interference for wireless traffic at different txrates. By shaping to 22Mbps, the weak TCP flow was still getting an unfair percentage of the total throughput. But, VOIDShaper knows that based on a previous iteration, the weak flow only achieved a tiny speed increase, and hence in this case, knows the weak flow is sending at a lower txrate. Network utilization is more important here, rather than complete fairness, since the weak flow cannot send faster than 5.5Mbps. Results were also shown if VOIDShaper continued shaping down towards the naïve rate of 4Mbps, which will be unfair to the strong sender.

4.7 Summary

The data presented allows us to fully understand the effects of shaping, and how VOIDShaper can have an impact on two wireless nodes sending at same and different txrates.

Let’s answer the questions set out at the beginning of this chapter. VOIDShaper was able to detect interference with traffic at symmetric txrates with ease, once the Emulab environment was setup with constant interference (which was not easy). The correlation data for traffic at different txrates looked similar to the symmetric txrate traffic, hence the txrates cannot be inferred from VOID output. Before and after shaping, VOIDShaper reported the correlation value as negative numbers, otherwise “None” if no interference is found. But, with the approach to shape TCP flows together, the correlation values still showed negative numbers, and did not indicate whether interference went away completely. This is because the traffic is shaped through a fair queue - VOID correlation becomes almost perfect (in the range of [0.95,1.05]) and hence masking the interference.
The shaping rate, if it needs to be set, should be at 85% of the total bandwidth of the interfering flows. At that point, the TCP flows were found to be sharing fairly, unless the weak sender was sending at a lower txrate, such as in experiment 4.6. A case where interference is detected but no shaping was necessary is shown in experiment 4.4. Shaping was only effective for traffic of different txrates where the strong sender was sending at a lower rate. Otherwise, a weak slow sender would drag all the fast senders down if VOIDShaper shaped to the slowest throughput. And, finally, the decisions made by the proposed traffic shaping algorithm is shown to be effective in shaping away wireless interference, except in the last experiment, where wireless fairness could not be achieved, overall high network utilization was preserved.
Chapter 5: Conclusion and future work

5.1 Conclusion
The VOIDShaper engine has shown to effectively find interference and shape the traffic for 802.11 devices sending at the same transmission rate. In this case, by shaping to 85% of the total bandwidth of the interfering flows, the strong flow and weak flow share the bandwidth equally and fairness can be achieved.

For nodes sending at different transmission rates, the cost-effectiveness of shaping was evaluated as VOIDShaper only knows the TCP flow throughputs, and not the actual transmission rate. The key is to look at the historical data across past VOIDShaper iterations. By knowing that interference exists, and whether traffic shaping has improved the TCP flow throughput, a shaping rate can be enforced to successfully achieve some level of fairness while not sacrificing network utilization. In the case of a strong sender sending at a lower rate, fairness was achieved the same way as symmetric txrate traffic. In the other case, a weak slow sender would not experience a significantly higher throughput even with shaping, as the txrate limits the maximum throughput.

5.2 Limitations and future work
VOIDShaper has been shown to work only with 4 wireless nodes, with two senders and receivers - this means only two TCP flows are analyzed and shaped together. VOIDShaper requires the traffic to be shaped through a central router, which may require a powerful router for larger networks in handling connections from hundreds or thousands of wireless clients. The wypy portion takes the longest to run at the moment, and can benefit from a redesign for scalability. Also, this has not been evaluated in live networks outside of emulab.

Future work includes the evaluation of the VOIDShaper algorithm on handling more wireless nodes (3 or more TCP flows). With more TCP flows, grouping of the interfering should be handled differently, with shaping actions applied independently to each group. The algorithm would also have to account for multiple groups.
Further policies for choosing the shaping rate can be considered, especially with more nodes involved - 85% may not be optimal for more competing flows. The policy regarding the removal of the shaping rules (mentioned in section 3.7) needs an in-depth look. What is achieved in this work is shaping to alleviate wireless unfairness. Once fairness is observed, at what point should the shaping rules be removed or modified? Will removing the shaping rules allow interference to come back? Lastly, speed improvements can be made in parts of the VOIDShaper pipeline to make it more real-time.
Bibliography


[17] Cai, Kan and Wang, Junfang and Lotun, Reza and Feeley, Michael J. and Blackstock, Michael and Krasic, Charles, "A wired router can eliminate 802.11 unfairness, but it's


[26] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac

http://www.python.org/

http://www.wireshark.org/


http://lartc.org/howto/lartc.qdisc.classless.html#LARTC.SFQ
Appendices

Appendix A

D-ITG command used to send traffic from client:

- D-ITG-2.6.1d/bin/ITGSend -a nodew2 -T TCP -t 3600000 -c 1448 -C 2590

Where:

- `-a` is the target computer which is running ITGRecv
- `-T` denotes TCP protocol in use
- `-t 3600000` denotes to send for 1000 minutes (long enough to conduct experiment)
- `-c 1448` denotes the TCP packet payload size of 1448 bytes (the largest packet without fragmentation)
- `-C 2590` denotes the number of packets to send per second

This amounts to a total of 1448 bytes * 8 bits/byte * 2590 ≈ 30Mbps

D-ITG command used to receive TCP traffic on wireless clients such as nodew2 and nodew4:

- D-ITG-2.6.1d/bin/ITGRecv
Appendix B

Traffic shaping commands used in Linux, where
$IF$ is the network interface,
$LIMIT$ is the max bandwidth

```bash
sudo tc qdisc add dev $IF root handle 1: htb default 30
sudo tc class add dev $IF parent 1: classid 1:1 htb rate $LIMIT ceil $LIMIT

sudo tc qdisc add dev $IF parent 1:1 handle 10: sfq perturb 10

sudo tc filter add dev $IF protocol ip parent 1:0 prio 1 u32 match ip dst 10.1.4.3/32 flowid 1:1   # nodew2

sudo tc filter add dev $IF protocol ip parent 1:0 prio 1 u32 match ip dst 10.1.1.3/32 flowid 1:1   # nodew4
```
Appendix C

Emulab hidden terminal topology full NS source file:

source tb_compat.tcl
set ns [new Simulator]

# Allocate the nodes. Their "wifi-ness" is determined later, # by the type of networks you request to be set up on them.
set nodew1 [ns node]
set nodew2 [ns node]
set nodew3 [ns node]
set nodew4 [ns node]
set router [ns node]
set client [ns node]

tb-set-hardware $client pc3000
tb-set-hardware $router pc3000

# A wireless lan connecting the first three nodes.
set lan0 [ns make-lan "$nodew1 $nodew2" 54Mb 0ms]
set lan1 [ns make-lan "$nodew3 $nodew4" 54Mb 0ms]
set wiredlan0 [ns make-lan "$router $nodew1 $nodew3" 100Mb 0ms]
set wiredlan1 [ns make-lan "$router $client" 100Mb 0ms]

# Choose the wireless lan protocol.
tb-set-lan-protocol $lan0 "80211g"
tb-set-lan-protocol $lan1 "80211g"

# Set an access point. This node becomes the access point;
# others in the LAN become stations of it. You can also set other # modes for your LAN, such as Adhoc mode.
tb-set-lan-accesspoint $lan0 $nodew1
tb-set-lan-accesspoint $lan1 $nodew3

tb-set-lan-setting $lan0 "mode" "Managed"
tb-set-lan-setting $lan0 "channel" 1
tb-set-lan-setting $lan0 "rate" 36M

# This can be set so strong flow sender isn't overwhelming the weak sender
#tb-set-lan-setting $lan0 "txpower" 1

# Select a wireless-capable image (i.e., FC4-WIRELESS)
# Let the other node default to RHL-STD (currently 9.0).
tb-set-node-os $nodew1 FC4-WIRELESS
tb-set-node-os $nodew2 FC4-WIRELESS
tb-set-node-os $nodew3 FC4-WIRELESS
tb-set-node-os $nodew4 FC4-WIRELESS
tb-set-node-os $router F8_IPTACCT
tb-set-node-os $client F8_IPTACCT

# Turn on Manual routing.
$ns rtproto Static
$ns run