End-to-End Performance Analysis and Scalability of Tablets

M.Tech Dissertation Report

Submitted for the fulfillment of requirements for the degree of

Master of Technology

by

Deepak Jayanth P M
Roll No: 10305913

under the guidance of

Dr. Deepak.B Phatak

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Mumbai
## Contents

1 **Introduction**  
1.1 Motivation .................................................. 1  
1.2 Scenario Considered .......................................... 1  
1.3 Sources of Bottlenecks ...................................... 2  
  1.3.1 Queueing systems[1] .................................. 2  
  1.3.2 Difficulties & Limitations ............................. 3  
1.4 Report Outline ................................................ 3  

2 **Literature Survey** ............................................ 4  
2.1 Performance Analysis ........................................ 4  
  2.1.1 Wireless Networks Basic Operation .................... 5  
  2.1.2 Analytical Methods .................................... 5  
  2.1.3 Software Methods ..................................... 9  
2.2 Scalability .................................................... 13  
  2.2.1 Scalability at Data Link layer ......................... 13  
  2.2.2 Traffic Control and Shaping ............................ 15  
  2.2.3 Multicast ............................................... 17  
  2.2.4 Protocol Customization ................................. 18  
  2.2.5 Comparison of Bottlenecks and Relevant Solutions .... 20  

3 **Analysis from Clicker Experiment Case Study** .............. 21  
3.1 Experiment Scenario ......................................... 21  
3.2 Number of Users connected vs Time ........................ 22  
3.3 Link layer Retransmission vs TCP layer Retransmission .... 22  
3.4 Goodput, TCP and Datalink overheads vs Time ............... 23  
3.5 Significance of Link layer Multicast, Broadcast .......... 24  
3.6 Number of Network Transactions per second ................ 26  
3.7 Number of TCP Connections Established per second ........ 27  
3.8 Wasted Bandwidth for outside connections .................. 27  

4 **Experimental Results** ....................................... 29  
4.1 Wireless Router Load Testing Tool .......................... 29
4.1.1 Purpose of this tool .............................................. 29
4.1.2 Algorithmic steps .............................................. 29
4.1.3 Testing .......................................................... 30
4.2 Minimizing Data ................................................... 30
4.2.1 Websockets ...................................................... 31
4.2.2 Cache and Compression ....................................... 31
4.3 Bandwidth Guarantees using Qos ................................. 32
4.3.1 Traffic Shaping on 4users at 8mbps ......................... 33
4.3.2 Traffic Shaping on 9users at 16mbps ....................... 34
4.3.3 Differential Traffic Shaping on 7users at 8mbps and ssh user at 16mbps 34
4.4 Laptop as Access Point .......................................... 34
4.5 Proposed Solution for large number of users .................. 35
4.5.1 Proposed solution for efficient Quiz ....................... 37
5 Conclusions ............................................................ 38
6 Future Work ........................................................... 39
Appendices ............................................................... 43
A Requirements for Running Load Test Tool ....................... 44
A.1 Installing linux wireless driver patches ....................... 44
A.2 Testing for frame injection support ............................ 45
B Graphs on All 3 Channels for Comparison ...................... 46
B.1 Goodput, TCP and Datalink adjustments ..................... 46
C Traffic Shaping Configurations .................................... 50
C.1 Experiment 1 Configuration .................................... 50
C.2 Experiment 2 Configuration .................................... 50
C.3 Experiment 3 Configuration .................................... 50
List of Figures

1.1 Scenario 1 ......................................................... 2
1.2 Scenario 2 ......................................................... 2
1.3 Potential sources of bottlenecks during a communication ................. 2

2.1 Hidden markov model for contention process ................................. 6
2.2 Time taken for transmission under Basic mode ................................. 6
2.3 Time taken for transmission under RTS/CTS ................................ 6
2.4 Throughput vs Transition probability under Basic mode [2] ................. 7
2.5 Throughput vs Transition probability under RTS/CTS [2] .................... 7
2.6 Wasted channel time vs number of stations[2] ................................ 8
2.7 Throughput vs Mean dataframe delay [3] ....................................... 9
2.8 802.11b/g A.P frequency planning ............................................. 14
2.9 802.11a A.P frequency planning ................................................ 14
2.10 A Pictorial representation of Network Traffic Shaping ...................... 16

3.1 No of Distinct IP’s communicating per second ............................... 23
3.2 TCP data from Server to Client through A.P ................................ 24
3.3 Goodput data ......................................................... 25
3.4 Goodput + TCP overheads .................................................... 25
3.5 Goodput + TCP overhead + Datalink overhead ................................. 25
3.6 UDP unicast data from Server to Client through A.P ......................... 26
3.7 UDP multicast data from Server to Client through A.P ...................... 26
3.8 The number of connections seen on the wireless network .................. 27
3.9 The number of TCP connection established .................................... 27
3.10 The bandwidth wasted by accessing outside connections .................. 28

4.1 Virtual connections made to an A.P ......................................... 31
4.2 Communication overhead of websocket ....................................... 32
4.3 Traffic Shaping Experiment 1 ............................................... 33
4.4 Traffic Shaping Experiment 2 ............................................... 33
4.5 Traffic Shaping Experiment 3 ............................................... 33
4.6 Traffic Shaping 4users at 2mbps ............................................. 33
4.7 Traffic Shaping 9users at 8mbps ............................................. 34
4.8 Traffic Shaping 7users at 8mbps and ssh user at 16mbps .................. 35
List of Tables

2.1 Comparision of bottlenecks and possible solutions ................. 20
3.1 Quiz Client Server Communication model ........................... 22
4.1 Scalable Quiz Model ............................................. 36
Abstract

The growing popularity of wireless devices such as tablets, smartphones demands more advancement in the wireless LAN technologies. Although these wireless routers are rated at a throughput of 54Mbps, 100Mbps, etc., the maximum throughput observed is typically of range 20 to 30 Mbps and it keeps on decreasing as the number of connected people to it increases. Wireless classrooms applications are limited simply by the wireless router’s speed that can be guaranteed to simultaneously connected wireless users. This work identifies the bottlenecks in connecting large number of wireless users (typically of 100+) in situations such as wireless classrooms to a single access point. And it also examines possible approaches to reach that target of large number of connected users. The paper examines the possible improvements of scalability at datalink, transport and application layer.
Chapter 1

Introduction

Performance is the most important characteristic of any network, service, application, etc. And the study of performance analysis is very important, as these study will help in making critical decision on particular hardware to use, technologies to implement, number of users to be served, predict the trend of future inputs to system, etc., or make new better technologies. Poor capacity planning and performance analysis will lead to improper functioning and high maintenance cost of the infrastructure. There has always been continuous advancements in wireless technologies in terms of speed. But when it comes to scalability, the wireless networks doesn’t always keep up with wired networks. This is for the simple fact that each connection needs to share a channel. Most of the Wireless routers deployed today still use 2.4Ghz which supports only 3 non-overlapping channels. The switch over to 5ghz which supports 23 non-overlapping channels may still take a long time because of cost issues. Hence the viable alternative is to use the existing technologies as efficiently as possible to achieve the target capacity. Making the target of large number of wirelessly connected users is possible only if the packets at all layers are handled efficiently.

1.1 Motivation

As tablets are low cost devices, it can replace future classrooms instead of books. Wireless classrooms require a constant guaranteed throughput to all connected users. Plans such as teacher less class rooms with a server distributing content to wireless users are simply limited by the capacity of wireless access points to deliver simultaneously to several wireless users. High end infrastructure cannot be deployed in rural environment mainly because of cost. So the cheap alternative is to use the existing low cost wireless routers efficiently to support large number of people.

1.2 Scenario Considered

The Figure 1.1 and 1.2 on page 2 shows the scenarios considered for which the analysis holds. Here ‘A’ indicates a wireless access point(A.P) connected to server ma-
1.3 Sources of Bottlenecks

In the scenario shown by Figure 1.1 and 1.2 on page 2 the sources of bottlenecks will be mostly at A and communication between A and C. This is mainly because each clients generate requests at very low rates, but when successive clients joint together the load altogether is exerted on access point only. For server any general purpose PC will be suitable as the loads less than 100 Mbps is not an issue over Ethernet. The bottleneck on wired hops of Figure 1.2 will vary based on number of hops.

1.3.1 Queueing systems[1]

The performance bottleneck can happen at many places other than the network. The following will bring an idea of places where bottleneck arise in general. Consider the following scenario, the queueing line represents potential sources of bottlenecks.
which can happen during a communication between a client and a server. When a client requests a connection, as shown in figure it undergoes the intermediate stages. The requests come in terms of packets to the NIC(Network Interface Card) of server and stored on a buffer as incoming request. The network buffer may already have similar kind of request packets obtained from other clients. This is possibly the first source of bottleneck, once this queue gets full the NIC wont be able to process further requests and it may become a source of bottleneck.

If the packet gets through the above queue, packet may next need to do task which requires the access of either of CPU , Database, IO. Each of these three (the CPU, Database , and Disk) have themselves queues of their own, and the same rule of NIC queue apply for the other three device queue. The only difference they are called with different names. At cpu level the bottleneck is caused because of different scheduling methods of process/threads queues. At Network level it because of packets arrived at buffers at Network layer. The packet can be even broken down and correlated with buffer size of incoming Frames on data link layer. The bottleneck queues can also be thought from the point of view of application design level, for eg, a chatty protocol, which sends too many short messages between peers. Here the percentage of bandwidth is wasted as overhead.

1.3.2 Difficulties & Limitations

The sources of bottlenecks can be anything from a network device, or host computer or a bad application design. This complex network of systems as a whole has a lot a components to study about. A single linux operating system has close to 100 tunable TCP and IP parameters[4]. So the way to measure performance and make sense out of the values obtained considering into account these many varying parameters is a very challenging task. And analysing this requires a careful understanding of each of such components.

1.4 Report Outline

The Following is the organisation of the report

- Chapter 1 Introduces to the topic
- Chapter 2 Literature survey on various performance analysis and scalability concerning wireless networks and its behavior with higher layers (Datalink,Network, Transport and Application)
- Chapter 3 Analysis from Clicker Experiment conducted with many wireless tablets.
- Chapter 4 Experimental Results
- Chapter 5,6 Conclusions and Future Scope
Chapter 2

Literature Survey

Most of the researches on wireless networks goes into improving speed. The speeds rated on A.P’s is of data link layer speeds. Typical speeds rated are 54Mbps (802.11g) and 150Mbps to 300Mbps (802.11n). But the speeds close to this can be only achieved only in controlled environment and mostly single user situation with no interference from bluetooth, microwaves, cordless phones which operate in same frequencies. Problems arise only when simultaneously wireless users try to connect. In those cases collision will happen. Collision is mitigated using exponential backoff at Medium Access Control of data link layer. Papers such as [2], [3] gives an analytical model for throughput at data link layer taking collision into consideration. Papers [5] discusses a model to identify the ideal number of users which can be supported gracefully in a single hop network based on given loads as inputs. The analytical model doesn’t consider many parameters to make the model simple. These assumptions may attribute to incorrect values in actual scenarios if models are not proper.

This section introduces to some of the ways of quantifying the performance of wireless systems in terms of analytical methods and software methods. It also discusses on scalability and ways of attaining scalability.

2.1 Performance Analysis

The section introduces to the ways in which the performance of the wireless networks is generally measured. Performance is usually measured using analytical, simulation and software methods. First is the analytical method which gives insight into performance, and finding out the bottlenecks in a mathematical way. On the Other hand, the performance Analysis of wireless devices done through software approaches are mostly done real time using tools in which performance analysis is done at application level. The third method is through simulation which is mostly similar to analytical methods with synthetic data and less mathematical.

Although analytical models has been approximated to some extent for keeping the model relatively small, they still provide a valid way to theoretically identifying the limits of a wireless networks and invidual devices. On the other hand the software approaches,
many of the inherent problems will be unable to measured since the details of complex underlying layers of protocols are not visible at application level. Application level performance analysis can be sometimes used by analytical models by constructing an approximated distribution corresponding to the application level measurement.

2.1.1 Wireless Networks Basic Operation

Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) is the access method on 802.11 Wireless Ethernet networks. A node wishing to transmit must first sense the medium, whether any other nodes are transmitting. If the medium is busy, the node waits for a period until the medium is free. If the medium is not busy for a short period known as the Interspace time the node tries to transmit the frame.

In infrastructure mode network each station needs a wireless access point act as a centralized hub. All wireless stations in an infrastructure mode network connect to an access point. All nodes connecting to the access point must have the same service set identifier (SSID) as the access point. In Ad-hoc mode network there is no need for an access point. All wireless nodes directly can interact with each other. All the nodes in an ad-hoc network must have the same channel and SSID. Although the medium access control (MAC) layer which is a sublayer of data link layer, throughout this paper both may be used interchangeably.

2.1.1.1 DCF mode

DCF defines a basic access method, and an optional four-way handshaking technique, with Request-To-Send/Clear-To-Send (RTS/CTS) mechanism. A wireless station with a packet to transmit, monitors the channel whether busy or free. If the channel is free for more than a slotted period of interval known as DIFS. The time immediately following an idle DIFS is slotted, and a station is allowed to transmit only at the beginning of each slot time. This duration is enough to for a station to identify whether other wireless stations are trying to transmit.

The DCF employs an exponential backoff algorithm to mitigate collision. The time after DIFS is slotted and each stations willing to transmit will choose a random slot inbetween 0 and w-1. Then it starts counting till that value. Once it reaches the stations transmit. Here, w is called the contention window which varies from Cmin to Cmax. If the transmission is unsuccessful the current value of w is doubled, and the stations again choose a random time between 0 to new value of w and then transmit. The counting time of a station is frozen if it senses the medium to be busy, the counting is resumed once again after a DIFS duration of channel not being busy.

2.1.2 Analytical Methods

The Analytical methods of performance analysis employ a bottom up approach of performance analysis. The analysis starts from modeling hardware layer functions and going
to higher layers such as Transport, Application etc. All the following analytical methods are ways of analysing from bottom to top i.e. starting from modeling the hardware and moving towards higher layers. The following analytical model quantifies the throughput obtainable at the medium access level (MAC) which is the sub layer of Data link layer. In other words, the following models roughly estimates throughput obtainable at datalink layer of wireless networks.

2.1.2.1 A model for throughput data link layer [2]

![Figure 2.1: Hidden markov model for contention process](image)

**Model construction** The behavior of a single wireless station is modeled using a Markov model. The exponential backoff process exhibited by every wireless station is modeled using two dimensional stochastic process with two random variables exhibiting a discrete markov chain. As the Figure 2.1 indicates each of the state of the markov chain represents two random variables, one is for representing the current backoff stage, and another is for counting the backoff timer value.

![Figure 2.2: Time taken for transmission under Basic mode](image)

**Duration for Transmission time** The duration of time in which the channel is busy is calculated using the following approach. The Figures 2.2 and 2.3 on page 6 shows

![Figure 2.3: Time taken for transmission under RTS/CTS](image)
the time spend for successful transmission as well as the time spend for collision under basic and four way handshake (RTS/CTS) mode as sensed by a wireless station. $T_{\text{success}}$ represents the time for a successful transmission of data by a wireless station. $T_{\text{collision}}$ represents the time spend by a channel to identify whether a collision is happened.

Both $T_{\text{success}}$ and $T_{\text{collision}}$ are expressed by the following equations.

$$T_{\text{success}}^{\text{basic}} = H + E[P] + SIFS + \delta + ACK + DIFS + \delta$$ \hspace{1cm} (2.1)

$$T_{\text{collision}}^{\text{basic}} = H + E[P] + DIFS + \delta$$ \hspace{1cm} (2.2)

$$T_{\text{success}}^{\text{RTS/CTS}} = RTS + SIFS + \delta + CTS + SIFS + T_{s}^{\text{basic}}$$ \hspace{1cm} (2.3)

$$T_{\text{collision}}^{\text{RTS/CTS}} = RTS + DIFS + \delta$$ \hspace{1cm} (2.4)

If $P_{tr}$ is the probability that there is at least one transmission in the considered slot time, $P_{s}$ is the probability that the transmission occurring on the channel is successful, $E[P]$ is average packet payload size and $\sigma$ is the duration of the empty slot time, then the throughput can be expressed as a fraction of average time spend by a packet of average size to the effective time spend in sending that packet, which includes time spend on collision, successful retransmission and the time of the empty slot.

$$S = \frac{E[\text{payload information transmitted in a slot time}]}{E[\text{length of a slot time}]}$$ \hspace{1cm} (2.5)

$$S = \frac{P_{s}P_{tr}E[P]}{(1-P_{tr})\sigma + P_{tr}P_{s}T_{s} + P_{tr}(1-P_{s})T_{c}}$$ \hspace{1cm} (2.6)

Figure 2.4: Throughput vs Transition probability under Basic mode [2] Figure 2.5: Throughput vs Transition probability under RTS/CTS [2]

**Observations** Figure 2.4 and 2.5 on page 7 shows the Throughput obtained vs the transmission probability($\tau$) of a wireless station. The transmission probability is the
probability that the given wireless station will transmit in a randomly chosen time after sensing a free DIFS duration time slot. It has been noted that this probability is the main factor of determining the throughput. This probability does not depend on the access mechanism. Given ‘n’ number of connected wireless stations, a lower value of $\tau$ means that very less number of wireless stations are interested to transmit. Higher values of $\tau$ means more number of nodes are interested to transmit. It is very clear from the graph that as $n$ goes higher and with a higher $\tau$ the system throughput decreases very fast. The graph can give a recommendation of values of $\tau$ each contenting wireless stations should have to achieve maximum throughput. Hence theoretically the maximum throughput can be obtained for any number of wireless users, but it cannot be practically achieved as this value is dependent on ‘m’ (maximum backoff stage) and ‘W’ (maximum backoff window) which are hardcoded in the physical layer and cannot be changed dynamically.

![Figure 2.6: Wasted channel time vs number of stations][2]

Figure 2.6 on page 8 is also another way to see the number of slots wasted, ie the number of time the channel is wasted on collision of data between one or more stations. If the initial backoff window size is very less, then for large values of ‘n’ more no of slots are wasted. For higher values of initial backoff window size, the no of wasted slots is less. The number of slots wasted by RTS/CTS mechanism is very low, which implies that DCF is more efficient in avoiding collisions for higher values of ‘n’ with RTS/CTS. The paper [2] discusses many other analysis on the above parameters under various circumstances.

2.1.2.2 Delay analysis at data link layer

The paper [3] shows the relation between the average delay experienced by wireless stations as the number of stations go higher. It uses a model which is an improvised model of [2]. As the number of stations go up throughputs increases and saturates at a point. The average mean delay experienced by stations to get a response back increases as the number of stations get higher because of collision. This situation is modelled in this paper and the following shows the result.

---

[2]: https://example.com/figure2.6.png
Figure 2.7 on page 9 shows the delay performances exhibited for different frame sizes. There is one more parameter which is not shown in the graph, which is number of stations. Three different data frame sizes (512, 434, 8184b bits) are considered for the simulation. From the comparison of basic and RTS/CTS mode, it is noted that the larger data frame offers higher throughputs. It is also noted that the RTS/CTS method offers more throughput than that of basic mode of operation. This is mainly because of the collision overhead in basic mode of operation. In basic mode the collision can be detected only after the whole payload has been transmitted. Since this overhead is not present in RTS/CTS mode, the collisions are less. This implies that the average delay due to RTS/CTS mode is lesser than that of basic mode. But it is also noted that in some cases the basic mode has higher throughputs. This can be attributed to the cases for smaller data frame (in this case for 512 bits) where the frame size is very small, such that the efficiency provided in detecting collision by the RTS/CTS mechanism itself has become an overhead. This implies that the overhead due to RTS/CTS is less prominent as the number of users gets higher and increases throughput.

By the use of Analytical models of performance it is clear that contention for wireless medium (through Distributed Coordination Function) theoretically limits the number of wireless users connected to a single access points.

2.1.3 Software Methods

The software approach employs a Top-Down approach of performance analysis. Analysis starts at Application Level and moved to hardware level. Since analytical methods always cannot model every thing, sometimes parameters measured through software approaches such as profiling, logging ,simulation can be used to approximate a parameter or distribution, etc for an analytical model. Also many times the analytical model may lose the details which are at application level. Like analytical models this method of analysis also gives some insight into predictions on the number of users supported, capacity of a network., etc. The software methods are normally employed when analytical models are very difficult to realize. They are useful whenever a distribution of a certain behaviors is difficult to model, but are easy to simulate or measure and can be matched with a known curve which is closely matching to measurements. This curve can be used as a distribution for certain arrival rate or service rate of an analytical model.
2.1.3.1 Benchmarking

Benchmarks give a relative measure of estimating the performance of any system and assigns a number to say how one machines scores high or low to another. The release of most of the new smartfones, tablets are usually followed by reviews about this new hardware’s performance over already existing hardware. But it a very abstract level of classifying a device. For some cases it may be sufficient, but in many cases this top level classification is not sufficient.

2.1.3.2 Android Application Profiling and Characterization [6]

The paper [6] examines ways of characterizing various application level of performance measurements. Here the performance analysis is carried from a single tablet/smartfone. This kind of analysis of performance analysis from a single device would provide a valuable solution to 1) app developers for better understanding of network operation and provide improved user experience, 2) the end users to understand strange behaviour of any application. The profiling used in this method to classify applications based on (a) app specification, (b) user interaction(swipe, click) (c) operating system IO (d) network. These details of analysis has other benefits such as to construct a models which closely resemble with the measurement performance parameter. These models can be then be used by analytical models, as analytical does not work with real datas as raw. They have to be approximated to close some known distribution. Some of the measurements made by [6] which are relevant to network communications are follows. All the measurements made on this paper are per single android device.

Background data

This paper[6] identifies discrepancies exist between the app specification and app execution of many selected paid and free version apps from android market. These apps are carefully choosen among 0.6 billion android apps based on number of downloads, ratings given by users, etc so as to represent a wide usage by many users. By the analysis they have found that free versions of apps could end up costing more(data usage costs) to the users than their paid counterparts, due to an traffic differences between free version and paid version. The methods identified that most of the network traffic is not encrypted. It is also noted that some apps communicate with more than 13 different sources for some operation with most of them being content distribution networks (CDN).

This implies that there is always a constant background data communications going on android devices. Since in android devices each applications run of a separate sandbox, its is not possible to turn off all these background communications when we need to run an application without rooting the device.

2.1.3.3 Traffic Characterization

The paper [7] measures how android devices behave collectively. Especially it examines the network traffic generated by a group of 33 android phones over a period of time.
Protocol Overheads  The overhead of lower layer protocols can be high based on the traffic. The experiments done in [7] of a dataset of 33 android phones suggests that the overhead due to lower layer protocol adds to 12% for regular traffic and 40% for traffic with Transport layer securities(SSL).

Network Performance  The network performance is characterized using TCP throughput, retransmission rate, DNS lookup time, Round Trip Time (RTT), latency. Since most network applications use TCP, TCP is an important performance metric of analysis. A request for a web object is initially done through a local DNS lookup, and the establishment of TCP handshake, Hence these two factors contribute heavily to user-perceived performance of many network applications. The RTT and TCP retransmission rate are calculated by matching each TCP packet with its Acknowledged packet. In case of retransmission the RTT for that packet should not be taken into account.

Browsing Performance  The Webbrowsing performance is a little complex because of the dynamic nature of content. Some of the metrics measured are TCP handshake time, TCP transfer time, DNS lookup time, Page loading time Content types, sizes and Javascript execution time etc., Some of the performance metrics like Page loading time which can be calculated by time it starts from a DNS lookup packet to the last connection data served of that request. This is slightly user-perceived measure of page loading time since even after the last data object delivery to client, the client needs to parse and run Javascript functions and render a page hence adding a significant amount of rendering time.

Summary  The user perceived performance measured using this paper is very important. The number of seconds each device wait before getting the data is one such user perceived performance.

2.1.3.4 Reducing Application load

The experimental results from [8] suggests some interesting analysis on packet losses. For small packet loss rate, e.g.,2%, there is little performance degradation. However, with 10% loss rate, the loading time rises up significantly.

Compression can significantly reduce web content size. For text objects, such as HTML, CSS, Javascript, PHP, etc., the object size can be reduced by around 70%. It shows that the content size can be reduced by more than 50% for most of the URLs. While compression reduces the bytes of data transferred over the network, decompression will increase computation overhead on the phone.

2.1.3.5 Performance Measurement Tools

Lack of Effective tools  There are lot of tools available such as Jmeter, httpperf, wireshark to analyse performance by means of logging of request datas. These logs ranges from application level information logs, transport level logs such as TCP, Network level logs such
as IP packets. The main problem with the tools which are available is that they are too much verbose. The Dumps which are produced is typically very large even for a 1 minute interval of logging. Almost all of these tools produce graphs to compare two or more related fields. Although Graphical interface exist they are still not enough to represent in a manner which is easily perceivable. Some of these tools which exist today can be only used by experts and only after a deep knowlegde of usage in the tool. As in case of most of the tools a single tool is not robust enough to diagnose a problem such a scalability.

Although tools such as wireshark exist to analyse all layers from linux environment, the same doesnt exist to analyse the operation of live access point, hence it is difficult to analyse and characterize operation at Data link layer of wireless router such as channels, contenting stations, data link layer packets, MAC operation, etc., Hence these operation are usually analysed only through simulation. The use of open source linux wireless firmwares such as provided by open wrt may be possible to some extent.

2.1.3.6 Issues in measurements

Issues Specific to tablets Although a variety of open source tools exist to measure performance from a pc, the opensource tools exist to measure from android devices are very less. The variety of testing which can be done from these devices is also less. For example tools such as Jmeter exist for generating customized test cases from PC, similar opensource tools of generating test cases of such high numbers are not available for android. Measurement tools from android is only useful when android tablet itself becomes the server for sending data. But in wireless classroom environments the load generated by a tablet is very minimum, the bottleneck will be wireless access point.

Other issues specific to tablet is these energy contrained devices turn on and turn off their receivers inorder to save energy. This is done by the devices by sending a power saving poll(management frame) to the access point, which indicates whether it is going to turn off or going to stay awake once in a while. Typical observations from wireshark indicates that the sleep duration ranges from 1sec to 5sec. The access point accordingly makes decisions on when to send a particular data for that machine. This behavior only important if all the datas are multicast or broadcast data. The access points adds this information on its beacons by means of Delivery traffic indication message(DTIM) interval, which says at what interval it will transmit multicast packets. The wireless devices which are sleeping can wake at these durations to receive that multicast or broadcast datas. In case of TCP data this behavior is not relavant.

Mapping Problems When ever anything is measured problem of logging overheads exists, if we record any network connections from the same machine or device the url is loaded. Since every packet is logged, it will increase the disk overhead. The maximum throughput measured will be inaccurate, since the systems utilization will rise quickly as more and more packets are coming. Which leads to more and more logging of datas. So there is an inherent bottleneck to the device as it is doing an IO operation extra by saving the dump. So at high rates of incoming request the device which does recording
will perform and serve less when compared to device which doesn’t do recording.

An alternative method is to record the connections from a third machine operating in promiscuous mode (wired devices) or monitor mode (in case of wireless devices). It will get all the input requests as if it were forwarded to it, and store all the records of data. This method seems to be good, but even this also has some inherent problems. There are always packets which get missed on the air due to collision, and SYN signal are resent by a request machine or ACK gets missed out in the air, the device might send the ACK signal again. In this method since the packets are not addressed to this sniffer machine, it cannot account either for the lost SYN signal, ACK signal or resent SYN, ACK signal. It simply assumes that each of SYN, ACK pair are new connection. As this is a wireless sniffer, it can itself miss out SYN, ACK signals.

Delayed acknowledgments can cause other problems such as measuring the Round Trip time (RTT) which is the measure of time between SYN and SYN-ACK packets. Because of delayed acknowledgments it is quite difficult to identify the data and its corresponding Acknowledgement.

Hence there seems to be is always a trade of between different methods. But the problems inherent due to self overhead in measurement can also be generalised and classified to overhead of certain known percentage associated with any measurement.

### 2.2 Scalability

Scalability is one of the important characteristics of any system, which is the ability of the system to perform gracefully under increased load conditions. It is the ability of a hardware, infrastructure, application to perform gracefully when its size or operating function needs to be expanded. Its not only the function of just working properly on the new higher requirement system, but also to take full advantage of the hardware, infrastructure, underlying architecture. Scalability is talked in terms of scaling vertically or horizontally. Horizontal scaling as the name suggests expands horizontally inorder to increase its serving capacity (For example introducing a new computer to the pool would be considered horizontal scaling). Vertical scaling replaces the existing node with a high performing node to serve the growing needs (For example replacing RAM, upgrading to SSD hardisk, etc are considered vertical scalability). Usually horizontal scalability is considered more cheaper.

#### 2.2.1 Scalability at Data Link layer

The following are some of possible problems identified which contribute to scaling issues at Medium access level. For each identified problems possible solution to tackle the problem is also identified.
2.2.1.1 Channel Planning

The Figure 2.8 and 2.9 show how the Access points should be deployed to minimize channel overlaps. Here each cell represents a single Access Point (A.P). Each cell (access point) can connect to 20 or 30 wireless systems. For 802.11 b/g the frequency of operation is 2.4Ghz with only 3 non-overlapping channels. But for 802.11a the frequency of operation is 5Ghz and the number of non-overlapping channels is 12. The Figure shows how they can be arranged in a hexagonal cell manner to avoid channel overlapping. For 2.4Ghz operation, a cell’s interference with another cell using the same channel frequency is just 1 cell away. But for 5Ghz operation a cell’s interference with another cell using the same channel frequency is almost two cell away. Each cell can accommodate 20 to 30 users. By this method, the A.P’s can scale horizontally to connect practically countless number of users. Only thing is that the Access points should be interconnected to switches of backbone via wired connection such as ethernet.

There is still a small possibility of interference in terms of signals from wireless connected tablets/laptops. Since we can only control the signal strength of an A.P but not the wireless stations, the stations at the cell boundary of any cell can emit strong signal enough to interfere with next cell with same frequency at one cell distance from current cell, this may lead to hidden node problems. This is especially problematic in terms of 2.4Ghz band, since there is only single cell gap between cells of same frequency.

2.2.1.2 Bottlenecks at Data link layer

802.11 DCF operation From the analytical throughput models and simulation results of DCF operation, it is evident that as the number of wireless network tablets grows higher, the tablets contention for the wireless medium remains the main bottleneck. As a result the more number of collisions happen when the number of wifi devices connected to Access Point exceeds 50 to 60. The wireless network gets congested with packets, resulting in more errors, more retransmissions and a final congestion collapse. This is because of the wireless medium is Half-Duplex and only one tablet can receive or transmit data at a time. A tablet cannot simultaneously listen and transmit. It can do only one operation at a
time. Although some new technologies exist where multi antenna support is provided, where one antenna can transmit and other can listen, but they are not largely available and still large number of users use age old standards. As the number of tablets increase the probability of collisions go larger, and the probability of the wireless medium to transmit any data without collisions is very less. Thus theoretically this defines a maximum limit of the number of wireless nodes connected simultaneously to a single wifi router.

Problem of Electromagnetic Interferes Since Wireless Networks works under frequencies which are publically available, there are a variety of other devices such as ZigBee devices, Bluetooth devices, microwave ovens, ISM band devices, security cameras, video senders, cordless phones operate or even other wireless routers under almost same frequencies close to 2.4Ghz, hence there is a huge probability in densely populated areas for the frequencies of other devices to overlap on the channels used. Hence for deployment for high capacity networks care should be taken such that it the wireless devices wont suffer electromagnetic interference from other devices using the same channel.

2.2.2 Traffic Control and Shaping

2.2.2.1 At Medium Access level

The high throughput/bandwidth measured doesn't ensure fair usage of channel because of the fact that in DCF it is possible for a single wireless node to hog the channel once it acquired it for a duration of burst of consecutive frames. A wireless device which is nearer will transmit at a higher rate than a distant device. The IEEE 802.11b/g/n doesn't ensure fair share of channel access. It is designed for maximum bandwidth. IEEE 802.11e has Quality of Service(QoS) support using priority queues, but this is available only in high end routers. This option may not be available to many routers available today. Even if it available on the router through flashing custom opensource firmware, due to the interoperatability standard the connected wireless cards also need to have this mode which is highly unlikely. Most users today still have wireless cards which supports only 802.11 b/g today. And there is no interoperatability standard exist between 802.11e with legacy systems.

2.2.2.2 At data link and higher levels

In a classroom scenario it is possible to increase the capacity of the wireless network by selectively discarding packets other than intended by teacher and student wireless devices through protocol examination. Wireshark captures shows that android device automatically try to connect to google servers or any other Content Delivery Networks(CDN) of applications installed on it. This leads to additional network traffic going on for android devices. Network traffic such as these must be identified and discarded.

There are three possible ways to implementing it, one at the router level where routers are configured to ditch packets other than that indented by teacher device to students. Next is at the server level, where the server has traffic control and ip firewall to drop
other connections. But both of this method doesn’t completely improve the situation. Because the student machine which is connected may still request for such data again and again which is still a overhead. The only overhead solved by this method is not to allow that some particular type to be downloaded. But still it doesn’t restrict the machine at medium access level. The request of such data is still a little overhead at connectivity level. The final method is restricting at the client device level, in this case the device needs to be rooted to add any such traffic rules.

One possible way to implement traffic control on the server machine is by using “tc command” (which is installed on linux machines by default). The tc command provides qdiscs (which are basically queues), classes and filters. The manual here [9] provides the list of all traffic control rules which can be implemented.

The following are some of the parameters of the tc command

- qdisc - qdisc are queues which says how the enqueue and dequeue of packets should happen. The different types of qdiscs are
  - fifo - First in First Out
  - sfq - Stochastic Fairness queue (Similar to round robin)
  - tbf - Token bucket filter
  - htb - Hierarchical token bucket
  - cbq - Class Based Queuing
- class - They are flexible components which can have multiple sub class components
- filter - Filters acts as a classifier to selectively let pass through or drop incoming packets.
2.2.3 Multicast

Although the support for multicasting exist since almost two decades, it is not widely used. One reason is because of the complexity involved in configuration and it is still a mystery for many ordinary users, network administrators. The difficulties lie in the implementation of routing tables to support the multicast operation which is not supported mode of operation of routers in general.

IP Multicasting is supported by group addressing where a source doesn’t need to know the ip address of the receiver, clients which needs to be associated with a group by means of IGMP protocol. Packets are delivered to receivers that have declared their membership in the group over a tree that spans all such receivers. The challenge lies in construction of the path tree on each routers. Whenever new devices joins or leaves a group this information has to be updated with path tree. This tree can itself become an overhead when the number of routers increases. IP Multicast is suitable if there are more than one than one hop between source to destination.

2.2.3.1 Single Hop Network

In single hop networks there is no need of intermediate routers, hence there is no need to construct the tree updating multicast information. Hence in case of server connected to client just by a single router, it is possible without any modification to the router.

2.2.3.2 Multiple Hop Network

On multiple hop networks, ie the client and server are connected by a number of interconnected machines, it is difficult to realize multicast. This is because each intermediate routers doesn’t know how to handle multicast data. Each intermediate routers needs to have trees constructed representing how to forward multicast packets.

There are a lot of open source routing software available, this can be installed on a stand alone server. Some of the opensource routing software available are Babel, B.A.T.M.A.N., Bird Internet routing daemon, OpenBGPD, Quagga, XORP, GNU Zebra, Vyatta. The server installed with this opensource router software should then be configured in a mode to provide support for building multicast trees on it through Internet Group Management Protocol(IGMP). The servers should be coupled with the router flashed with an OpenWrt firmware. Any Routers on the path between source and client which needs to support multicast needs to be replaced with this (Server with open source routing software + OpenWrt firmware flashed routers) combination.

If end user computer is behind NAT, it can not subscribe to multicast directly, so it needs some kind of proxy to do it for him. OpenWRT comes with igmpproxy utility to provide support automatically. A method of implementation of this technique is in detail is discussed in the paper[10].
2.2.4 Protocol Customization

Tweaking or tuning is the process of modifying the existing system, application, protocol to get the maximum performance out of it. While it is not always done to improve only performance some times it can be done to accomplish a specific need. The following is one such method of modification of the protocol to suit the needs. The challenge lies in ensuring the compatibility of the protocol after modification to operate along similar devices. Other challenges include keeping up to date with different future versions of devices.

2.2.4.1 At Data link layer

Overview of 802.11 frames  The 802.11 frames are classified into three types of frames management frames, control frames and data frames. Under normal operation each wireless client’s network interface card handles all these frames and only shows data which are intended for the particular device to the higher layers. Hence all management frames such as beacon frame, association request, association response, disassociation, authentication, deauthentication and control frames such as Request to Send (RTS), Clear to Send (CTS), acknowledgments, etc are hidden to the other layers under the normal operation. They are filtered by the wireless card on the clients. There is a special mode which a wireless device can operate which is called monitor mode. This mode is not supported by some cards. Under monitor mode, none of the management and control frame are filtered. This mode is useful to see what’s going on on the network. It is possible to have the wireless card in both monitor mode and normal mode by means of the mac80211 framework for linux.

When wireless devices are under this mode, it can see all the management and control frames which are not meant for it. The main idea is to exploit this method, and send datas along any one of this management frames and control frames. Considering a single server and multiple number of client model, by this method it is possible to send data by a server machine to all the clients at once, since all data is visible to everyone without even the clients actually connecting to an Access point.

MAC Broadcast using Beacon Stuffing[11]  This paper [11] discusses a method of low bandwidth communication protocol for IEEE 802.11 networks that enables APs to communicate with clients without association. The idea is not for increasing the throughput but using this modification to support a low bandwidth broadcast transmission to all connected wireless devices from an access point. This is done by overloading 802.11 management frames with data while still maintaining the interoperatibility of the protocol. The beacon stuffing protocol is based on two main operation of wireless devices. First, the wireless clients receive a particular access point’s beacon signals even without associating with it. Second, it is possible to overload management frame with data.
**Mechanism**  This approach is based on the push model of information delivery. The key idea is to overload IEEE 802.11 beacons to carry additional information. Beacon frames are used to announce the presence of a Wi-Fi network. As a result, an 802.11 client receives the beacons sent from all nearby APs, even when it is not connected to any network. In fact, even when a client is connected to a specific AP, it periodically scans all the channels to receive beacons from other nearby APs to keep track of networks in its vicinity. The client does not have to transmit anything to receive the beacons; it merely has to listen. This push model is in contrast to the model currently being used where a client establishes an Internet connection, transmits information about its location (obtained in a variety of ways), and pulls information relevant to that location. Under common environmental conditions, the beacons frames have a range of 100-200 meters.

All the information are treated as broadcast string of data bytes. The data can be anything ranging from text, audio, video, etc. only main thing is at the client side this information has to be ordered properly to make sense out of the data.

**Opensource linux wireless drivers**  There are several opensource drivers available for linux wireless devices. Although drivers are available as opensource for most of the devices the firmwares are available only in a binary form. Madwifi[12] is project which provides opensource Linux kernel drivers for Wireless LAN devices of Atheros wireless chipsets. MadWifi is one of the most advanced WLAN drivers available for Linux today. It is stable and has an established userbase. The driver itself is open source but depends on the proprietary Hardware Abstraction Layer (HAL) that is available in binary form only. ath5k and ath9k are newer version of this driver. ath9k is the youngest of the three drivers. Initial development was done by Atheros, who then released the complete source code to the community. ath9k supports all currently available 802.11n chipsets from Atheros. Many Realtek chipsets has complete opensource code available. Intel based cards use proprietary firmware.

**2.2.4.2 At Network level**

For many of the scalability issues identified almost all of the solutions requires some form of customization to support the operation stated. Customization modify cheap existing hardwares to make more out of it. But all existing wireless routers and wireless cards cannot be customized because it is mainly controlled by vendors. Most of the vendors dont want to make source for firmware open, in the fear of losing their market share to new growing company. For some of the hardwares available even if the code is opensource, some part of the code may use some form binary Hardware Abstraction Layer(HAL). Hence making this HAL unmodifiable eventhough its under opensource as a whole entity. The following methods are some of the methods to achieve some levels of customization.

OpenWrt is a Linux distribution for embedded devices which can be used to customize the commercial Access Points to get the most out of it. In a nutshell using openwrt one can get features only found in expensive enterprise solutions by installing its firmware on a cheap router. It can be used as a study tool to get hands on experience with 802.11

19
network layer and other higher layer operations rather than studying through simulations which sometimes doesn’t give exact results. OpenWrt supports many Routers available today.

### 2.2.5 Comparision of Bottlenecks and Relevant Solutions

The table 2.1 on page 20 summarises the techniques which can be employed in order to solve bottlenecks. All the solutions does not work for all the cases. But each of the solution contributes a little improvement which when combined together can work effectively.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Layer</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimizing application overhead</td>
<td>Application</td>
<td>Application Cache, Websockets, javascript compression</td>
</tr>
<tr>
<td>Minimizing Network overhead</td>
<td>Transport</td>
<td>UDP, Multicast</td>
</tr>
<tr>
<td>Bandwidth guarantees using Qos</td>
<td>Datalink, IP</td>
<td>Traffic shaping, policing</td>
</tr>
</tbody>
</table>

Table 2.1: Comparision of bottlenecks and possible solutions
Chapter 3

Analysis from Clicker Experiment Case Study

3.1 Experiment Scenario

Bottlenecks due to interference is not considered as it cannot be measured with a wireless device. Spectrum analysers are required to measure the effect of interference of neighboring bluetooth, microwaves or cordless phones on the throughput.

- Three access points where installed on channel 1, channel 6 and channel 11 respectively.
- The whole capture runs for approximately 40mins.
- The wireshark capture was a merged one of two individual captures.
- During first capture, approximately 90 tablets where connected to 3 access points (30 users per A.P).
- The next capture, approximately 160 tablets where connected to 3 access points (50-60 users per A.P).
- The gap in the graphs approximately between 1100 to 1200 seconds indicates the stopping of wireshark between two captures.
- During first capture, 2 quizzes where started and ended.
- During second capture 3 quizzes where started and ended.
- The capture was made using 3 USB wireless adapters working in monitor mode on 3 separate channels respectively.
- The tall peaks in the graphs represents quiz start or quiz end, where simultaneously all users try to connect
- The graphs shown in the sections are from channel 1, for other channel graphs see Appendix B
Quiz Client Server Communication Model

<table>
<thead>
<tr>
<th>Quiz Status</th>
<th>Request Method</th>
<th>URL</th>
<th>Reply</th>
<th>Periodicity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before Quiz Begin</td>
<td>GET</td>
<td>QuizSSE.jsp</td>
<td>Quiz not started</td>
<td>Every 1 second</td>
</tr>
<tr>
<td>During Quiz Start</td>
<td>GET</td>
<td>Quiz.jsp</td>
<td>Quiz data (json)</td>
<td>Once</td>
</tr>
<tr>
<td>During Quiz Running</td>
<td>GET</td>
<td>QuizTimeUpdate.jsp</td>
<td>Seconds remaining</td>
<td>Every 1 second</td>
</tr>
<tr>
<td>During Quiz End</td>
<td>POST</td>
<td>QuizResult.jsp</td>
<td>Acknowledgement</td>
<td>Once</td>
</tr>
<tr>
<td></td>
<td>GET</td>
<td>QuizResultSSE.jsp</td>
<td>Result XML</td>
<td>Once</td>
</tr>
<tr>
<td></td>
<td>GET</td>
<td>QuizListener.jsp</td>
<td>Redirects to QuizSSE.jsp</td>
<td>Once</td>
</tr>
<tr>
<td>After Quiz End</td>
<td>GET</td>
<td>QuizSSE.jsp again</td>
<td>Quiz not started</td>
<td>Every 1 second</td>
</tr>
</tbody>
</table>

Table 3.1: Quiz Client Server Communication model

3.2 Number of Users connected vs Time

The number of users connected is a slightly tricky term here. Because every user was connected to the access point all the time. One way to differentiate this is to have a condition that the number of distinct IP’s observed per some sample interval. Even after this condition, it’s quite difficult in this case. If the sample time is 1 sec, then the number of users connected is the number of distinct IP’s observed every 1 second. Here since every client is asking the server the time update every 1 second, the number of users will be almost constant. The figure 3.1 on page 23 shows the number of users connected vs time interval. Here there is a significant difference observed only between first and second capture. Other than that, throughout the y-axis is almost constant.

3.3 Link layer Retransmission vs TCP layer Retransmission

The fig. 3.2 on page 24 shows a typical TCP data connection from System 1 (Server) to System 2 (Client) through an Access point. In the figure each transaction is marked by an arrow with a caption on top of it. The arrows also end with a number. The TCP connection shown in this figure is after connection establishment. The RTS/CTS mechanism are turned on automatically or by setting on the router. In the fig. 3.2 if L2(ack)[indicated by 5] does not reach the A.P, the A.P sends the packet no. 4 again to
client. This is called link level retransmission. In the same way if packet no. 9 is lost, packet no. 8 is send to access point again. This is also link level retransmission. If packet number 10 doesn’t reach server then the server sends the data 1 again and follows the same procedure. This mechanism is called TCP retransmissions. The amount of time the device has to wait before TCP retransmission is calculated based on round trip time (RTT). The TCP layer retransmission is a well balanced mechanism which uses adaptive retransmission which calculates when to transmit a packet again, which is roughly twice as that of the time it waited for the first time retransmission.

A timeout period which is dynamically calculated is responsible for each retransmission. The timeouts are usually dynamic which is calculated from the Round Trip Time(RTT). This is according to the TCP adaptive retransmission specification. So after a packet is send, if no acknowledgement is received for the packet till the timeout period, the packet is retransmitted. This timeout period is calculated dynamically every time based on RTT. The RTT may vary based on number of hops, network conditions, etc. The timeout value varies exponentially with every failed retransmission in case of TCP.

In case of link layer retransmission, this exponential increase of the timeout value is not observed. This is well reflected in the graphs that the percentage change between the goodput and the datalink retransmission goes as high 500% as the number of wireless clients increase.

3.4 Goodput, TCP and Datalink overheads vs Time

This section shows how much of the bandwidth is wasted due to tcp retransmission, and link level retransmission. Three graphs are compared in this section. First is goodput, second is TCP adjustments and third is datalink adjustments. The following describes what the graphs denote.

- The fig. 3.3 here defines all TCP data excluding TCP retransmission and datalink retransmissions.
The fig. 3.4 here defines all TCP data including TCP retransmissions and excluding datalink retransmissions.

The fig. 3.5 here defines all TCP data including TCP retransmissions and datalink retransmissions.

In the three graphs fig. 3.3, fig. 3.4, fig. 3.5 the percentage change between goodput and tcp retransmission is very low, which indicates that the TCP’s adaptive retransmissions working very well. But the percentage change between the goodput and datalink retransmissions are very high, indicating that the link layer retransmission is not as robust as TCP layer acknowledgements.

3.5 Significance of Link layer Multicast, Broadcast

From the earlier goodput, tcp and datalink retransmission graphs its clear that, higher the number of wireless users the datalink retransmission will cause a lot of waste of bandwidth. An alternative suggestion comes from the following fact. Consider fig. 3.6 and fig. 3.7 on page 26. The UDP unicast packet does not have a transport layer acknowledgement as TCP unicast. And the UDP multicast doesn’t have either transport layer acknowledgement nor datalink layer acknowledgement. In classroom environments this has a significant advantage, since each of the listening clients needs the same data in classroom environments, instead of sending unicasted one to one connection, one to many.
can be sent and it will not require datalink acknowledgement. There is a possibility of packets received as errors, but there are udp protocols with a reliable error correction methods. The time at which a frame is retransmitted if no acknowledgement is received is uncontrollable. Its hardwired in the wireless devices. Hence when large number of
wireless devices are there, it is ideal for going to a protocol which does not have datalink acknowledgements.

Figure 3.6: UDP unicast data from Server to Client through A.P

Figure 3.7: UDP multicast data from Server to Client through A.P

3.6 Number of Network Transactions per second

The fig. 3.8 on page 27 indicate how many transactions are running per second are observed. The transaction here quantifies any flow of communication which happen in the wireless channel. This overhead is constant across the time. The values of 200 to 400 indicates the continuous GET requests send by the clients to update the time on the tablets and the continuous poll for asking the start of quiz time. This overhead can be eliminated using websockets, where the client need not poll the server continuously for data. It will automatically get the data from the server.
3.7 Number of TCP Connections Established per second

The fig. 3.9 on page 27 refers to the total number of TCP connections made throughout the capture process. As it is noted from the graph, there is always around 40 to 50 new connections established on the second capture. This is an overhead which can be minimized if keeping the connections open for a long time. This is possible with the help of long polling or better by using websockets.

3.8 Wasted Bandwidth for outside connections

The fig. 3.10 on page 28 shows the amount of bandwidth wasted by tablets communicating with outside machines starting file transfer downloads. From the wireshark data, it was observed that one tablet initiated some file transfer on downloading some dictionary file. This type of bandwidth wastage can be eliminated and has to be eliminated using traffic
control rules, which will either drop the packet or lower its rate of download speed. If many people initiates a background file transfer simultaneously during a quiz application run, even 10 people cannot connect properly to a wireless access point and download quiz.

The traffic control using traffic shaping experiments on the experiment section deals with this kind of situation. The traffic can be controlled based on the destination ports, source ports, based on protocols, based on ip, etc.

Figure 3.10: The bandwidth wasted by accessing outside connections
Chapter 4

Experimental Results

4.1 Wireless Router Load Testing Tool

Wireless access points are tested in controlled environments with no presence of interference from bluetooth devices, cordless phones, microwaves and interference from other access points on the same channel. The throughput rated using this is not very useful in actual conditions. Because in actual sites where it is deployed, there will be interference from all the cases mentioned above. Hence the throughput measured vary from area to area. So there should be tool which measures some throughput qualities based on the particular site where it is deployed.

4.1.1 Purpose of this tool

The throughput measured using traffic generators also doesn’t correlate much to actual conditions. This is because by generating traffic between a server and client machine connected through an access point, it doesn’t give idea about how multiple clients use the access point at same time. This multiple clients can be simulated using complex testbeds of wireless adapters, but that setup would be costly and not very easy to build and test flexibly at every location of wireless deployment.

The load test tool should actually load the access point to show how access points behaves when multiple clients connect. For example access points deauthenticate a clients, to serve new clients. Such behaviors can be understood by this tool. This tool is just another way of viewing traffic generation to access point.

4.1.2 Algorithmic steps

- Step 1: Using a single wireless adapter, tool should
  - Step 1.1: Send Probe Request and wait for Probe Response from Access Point(A.P)
  - Step 1.2: If 1.1 is successful, send Authentication and wait for successful Authentication reply from A.P
– Step 1.3: If 1.2 is successful, Send Association Request and wait for Association Response from A.P

• Step 2: If Step 1 is successful, do following steps
  – Step 2.1: send DHCP Discover and wait for DHCP Offer
  – Step 2.2: If 2.1 is successful, Now an IP is assigned corresponding to this mac id. Now send DHCP Request and wait for DHCP ACK from A.P

• Step 3: Change its mac address to a new one and do the Step no.1 to get a different IP address from the Access point.

• Step 4: Repeat the above steps to saturate the Access point by a single wireless adapter, there by mimicking different users to the access points, but in actual case all connections are made from a single wireless adapter.

• Step 5: If an de-authentication signal is received from the access point repeat step 1.2 and 1.3.

• Step 6: Send a load packet at every interval specified by load interval

4.1.3 Testing

The fig. 4.1 on page 31 shows that the tool is working to create virtual connections to wireless router. This is the router’s weblogin page showing the list of attached wireless devices to it.

The router assumes that the connections are coming from 3 different systems, but all where generated from a single machine. Device name starting with "SN 00*" are virtual connections made to the router from a single wireless adapter card. The device name "JAYDEE" is an actual connection to the access point with a valid mac address. Wireless actual point treats all connections alike and lists them all.

This tool needs further development to automate this completely. Currently the tool requires a little manual intervention to make a virtual connection by opening a separate terminal for each virtual connections.

4.2 Minimizing Data

The following section describes the ways in minimizing the protocol overheads and minimizing application loads using latest techniques available. The two subsections deals with reducing overheads at application level. First websockets provides a way to reduce HTTP overheads and next html5 cache reduces application load by caching information in the browsers for a period of time. Compression is a technique which is sometimes ignored, but can contribute significantly in case libraries in javascript.
4.2.1 Websockets

The following results are based on websocket implementation on nodejs given in the link [13]. The keepaliveInterval was changed to a large number before starting the experiment. This interval determines how long the client and the server wait before sending keep alive. If this value is made to very high value, it means that the connections can be kept alive that much time without sending any data. This is the main advantage when compared to regular HTTP where the connection will be closed immediately after no data is to be send. The fig. 4.2 on page 32 shows the number of bytes send by a chat application developed using websockets. Here, the initial spike represents the initial tcp handshake and HTTP upgradation by a websocket connection. the long flat line indicates that no chats are send at that interval. Very small peaks indicates the maximum bytes for sending message. No new connections are required for a websocket. The first TCP connection which was made during initial handshake is kept open and utilized as a bi directional socket.

- Typical HTTP communication data size: length(Appdata) + HTTP header of 300 to 400 bytes + (TCP,IP,Frame) of 60 bytes
- Typical Websockets communication data size: length(Appdata) + (TCP,IP,Frame) of 60 bytes

4.2.2 Cache and Compression

The other ways of minimizing data, is to minimize as much as possible in application layer. Caching and compression can be used whenever possible. A typical javascript jquery library comes around 250kB if uncompressed and 80kB if compressed. Application
cache can be used on html5 compatible browsers. In order to use it, a manifest[14] file needs to be specified which says which all files need to be cached and the rest to not cache. For example a typical web application, the javascript libraries and other such libraries whose code wont change, can be in the cache and rest of network related authentication can be not cached.

4.3 Bandwidth Guarantees using Qos

Classroom environments are different from file transfer situations. In such environments one system cannot hog the network. For example during a quiz conduction, if one wireless device start some file download, it may take up most of the entire bandwidth. The network bandwidth has to be shared equally. With already very less available bandwidth in case of wireless networks, it is extremely important to shape the network traffic if we have to support many simultaneously connected people.

For such cases traffic rules has it own plus and minus. The traffic shaping doesn’t always work as planned for all types of traffic. It is also possible for traffic shaping to worsen the already existing bandwidth thus defeating the purpose. The following graphs shows how traffic shaping can be implemented with examples.

The following experiments where conducted with a machine running DC++ and uploading to ‘n’ users. The DC++ is taken as a source of random inputs as the number of connected people vary with different latency, RTT,etc. The traffic rules are implemented using the linux tc command. The fig. 4.3,fig. 4.4,fig. 4.5 on page 33 shows the traffic control implemented pictorially. Here HTB implies hierarchial token bucket which allows only traffic rated at it to pass through and all others to discard. SFQ implies Stocastic Fairness Queue, in this queue every distinct connections passsthrough it gets a stochastic fairness rate. The way each packet should pass through gets fairness are decided by the stochastic queue by means of some hash functions, which gets refreshed every 10 seconds so as to not set any fixed priorities. On experiment three an additional queue called a priority queue is inserted to pass traffic through a higer 16mbps HTB class in cast of destination port equals 22(ssh port).
4.3.1 Traffic Shaping on 4users at 8mbps

The fig. 4.6 on page 33 is an excellent example where the traffic shaping worked perfectly as configured. Here the maximum rate of default queue which passes the traffic was set to 8mbps. In between the experiment the traffic control was reset and again implemented. Here all four machines share the 8mbps equally of 2mbps each. When traffic control was off, they traffic of all four went to random speeds. When traffic control was back on, the four users transfer rate came back once again to the specified traffic rate specification of under 8mbps.
4.3.2 Traffic Shaping on 9users at 16mbps

The fig. 4.7 on page 34 shows the same set of rules as the above except the traffic was limited to 16mbps. Here the traffic after controlling where not smooth as the previous experiment. There are a lot of fluctuations. But still the maximum upper limit set by the traffic control was followed. The fluctuations can be attributed to the fact the classification of the packets itself may become a bottleneck. As the classification specification gets more and more into the packets, each packets at the queue needs to be examined to classify them. The traffic rules also needs to be simple as possible.

![Figure 4.7: Traffic Shaping 9users at 8mbps](image)

4.3.3 Differential Traffic Shaping on 7users at 8mbps and ssh user at 16mbps

The fig. 4.8 on page 35 shows a differential setup where any ssh connections where given a maximum rate of 16mbps and all other connections where given a rate of 8mbps. Its clear from the figure that, the ssh connection utilized most of that 16mbps and the rest of all 7 users shared the 8mbps.

4.4 Laptop as Access Point

The output of this experiment was, to operate a laptop as an Access Point. The procedure given in [15] was followed. Normal Access Points(A.P) doesn’t give control over what A.P send to clients. There are some protocols which are initiated by the A.P which are only used only when there are multiple hops on network. There are some other protocols which are not widely used yet(such as IP version 6) but still initialized from A.P and clients.

So these contributes to the wasted bandwidth to the existing bandwidth. Even if these protocols are not initiated still the Wireless LAN will work as desired over a single hop wireless network. This is the initial goal of the experiment. According to the procedure given by [15], hostapd and dhcp needs to be installed as requirements. Laptop was able to
be operate as Access point. A software Access point running in infrastructure mode was created. This mode is possible only if the wlan card supports master mode of operation. The A.P was detected by all wireless devices. The implementation is done using hostapd and dhcp. Some protocols such as SSDP where not initialized in this version of hostapd, contributing to a lesser packets in the wireless network. SSDP is a service discovery protocol, access points send information about the services it provides to a multicast address on the network. Interested machines can get those services by sending M-search messages.

It is evident from Load testing tool that, the wireless router can be fooled into thinking of itself connecting to different users. The wireless load testing tool didn’t initiate typically multicast services. Still the whole network was working. This experiment is the reverse of load testing tool, that the clients are believed into thinking the laptop is an actual access point. Even though some services are not initiated, whole communication worked as intended without any problem. Both of these experiments are proof that the wireless networks can be customized to some extent to not initiate protocols, thus reducing bandwidth wasted for those protocols and still able to run the network as intended.

4.5 Proposed Solution for large number of users

The total available bandwidth varies based on the area where the access point is situated. The wireless load test tool will give an approximate upper bound to how many users, load the wireless access point can serve. This equals to the available bandwidth. After then a careful planning of per user bandwidth can be calculated. Traffic rules can be implemented based on this calculation, to guarantee bandwidth based on priorities. The traffic rules can be set such that the server machine has highest priority and higher available bandwidth. The clients will have comparatively lesser guaranteed bandwidth which they have to share with other clients.
<table>
<thead>
<tr>
<th>Quiz Status</th>
<th>Client Actions</th>
<th>ServerActions</th>
<th>Periodicity</th>
<th>No of Bytes (second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before Quiz Begin</td>
<td>GET Start.html</td>
<td>Cache javascript libraries, images,css</td>
<td>Once</td>
<td>len(jslibraries, css, images) + 1000</td>
</tr>
<tr>
<td>Before Quiz Begin</td>
<td>GET Auth.html</td>
<td>load auth.js</td>
<td>Once</td>
<td>len(auth.js) + 1000</td>
</tr>
<tr>
<td>Before Quiz Begin</td>
<td>Open WebSocket</td>
<td>Keep connection open</td>
<td>Once</td>
<td>1000</td>
</tr>
<tr>
<td>Quiz 1 Start</td>
<td>Ack</td>
<td>Send Quiz data</td>
<td>Once</td>
<td>len(quiz) + 60 + 60</td>
</tr>
<tr>
<td>Quiz 1 Running</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Quiz 1 End</td>
<td>Send answers</td>
<td>Ack</td>
<td>Once</td>
<td>len(answers) + 60 + 60</td>
</tr>
<tr>
<td></td>
<td>Ack</td>
<td>send Quiz 1 results (json)</td>
<td>Once</td>
<td>len(results) + 60 + 60</td>
</tr>
<tr>
<td></td>
<td>Ack</td>
<td>nextquiz = true</td>
<td>Once</td>
<td>len(nextquiz) + 60 + 60</td>
</tr>
<tr>
<td>Quiz 2 Start</td>
<td>Ack</td>
<td>Send Quiz data</td>
<td>Once</td>
<td>len(quiz) + 60 + 60</td>
</tr>
<tr>
<td>Quiz 2 Running</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Quiz 2 End</td>
<td>Send answers</td>
<td>Ack</td>
<td>Once</td>
<td>len(answers) + 60 + 60</td>
</tr>
<tr>
<td></td>
<td>Ack</td>
<td>send Quiz 2 results</td>
<td>Once</td>
<td>len(results) + 60 + 60</td>
</tr>
<tr>
<td></td>
<td>Ack</td>
<td>nextquiz = false</td>
<td>Once</td>
<td>len(nextquiz) + 60 + 60</td>
</tr>
<tr>
<td>After Quiz End</td>
<td>Close websocket</td>
<td>Ack</td>
<td>Once</td>
<td>60 + 60</td>
</tr>
</tbody>
</table>

Table 4.1: Scalable Quiz Model
4.5.1 Proposed solution for efficient Quiz

The table 4.1 on page 36 describes the requirements of a scalable quiz model. With websockets, html5 cache this can be achieved. As seen from the table, the websocket is held open always. The subsequent quiz can use the same connections to load data at just the overhead of 60 bytes of sending and 60 bytes for acknowledgement. Regular HTTP requests average header size is 300 to 400 bytes + additional 60 bytes of tcp header and the request HTTP header comes to 300 to 400 bytes + additional 60 bytes header. Total coming to 1000 bytes for every transaction. websockets totally eliminates the http header overhead of 800 bytes every connection. The next problem solved is the TCP connection establishment overhead everytime of (60 + 60 + 60 bytes) and connection termination overhead of (60 + 60 +60 bytes). The html5 cache saves images and javascript libraries which need not be downloaded every time. In this setup as soon as the clients establish the connection the total transport layer overhead for that client is 0 bytes until the quiz starts. Since all the quiz datas are server initiated, the server has entire control over how the data is send. It can send one by one or in groups thus decreasing the chance of collision by some amount. Collisions happen more when all clients at the same time wants to occupy the channel to send data. This doesn’t mean collision is eliminated. At datalink level each machines sends keep alive signals now and then to the access point. But it significantly decreased the collision due to client machines queriing http request and receiving http request at the same time. In this same setup, traffic shaping needs to be implemented in case of dropping any outside connections for file transfer,background data,etc are made from the devices.
Chapter 5

Conclusions

Scalability in wireless networks is usually achieved horizontally by deploying more access points. A single access point has its limits on the available bandwidth. The experiments and analysis suggest that, there is room for improvement in every levels of communication from datalink to application layer. The “wireless load test tool” and “laptop as access point” experiments suggest that a single device can play the role of client as well as access point. The amount of data transferred in both of the experiments are less than that of what normal access point and normal wireless client would transmit and still maintaining connectivity as it is supposed to. As the number of clients increases, the serving rate to each client should be decreased in order to keep up the connections with all clients. Applications should be designed based on one or more of the experiments analysed, to definitely see improvements on the number of connected people.
Chapter 6

Future Work

- Improving the wireless load testing tool to be robust to get more informations based on its load to the Access point.

- Currently the graphs in the analysis shown can be obtained by giving a wireshark capture file as input to a shell script. It is made to automate completely. The shell script runs other sub scripts to finally produce all the graph images. One particular future work could make use of live graphs to represent the analysis. This can be made as a plugin to wireshark.

- Building a Quiz application with complies with the improvements that can be done at all layers as suggested by this work.
Acknowledgements

It is my great pleasure and privilege to have Dr. Deepak B. Phatak, of the Department of Computer Science and Engineering as my guide and to take up the M.Tech Project under him. His inspiring guidance, experienced suggestions and dedication towards education keeps me motivated to do research work in technologies supporting education.

I also thank Mr. Nagesh Karmali for his valuable suggestions and fruitful discussions which kept me motivated.
Bibliography


Appendices
Appendix A

Requirements for Running Load Test Tool

A.1 Installing linux wireless driver patches

In order for the load test tool to work, the following patch must be applied to the linux wireless driver and the driver needs to be reinstalled.

Download any of the stable release of compact linux wireless from [16] according to the version of linux kernel of the OS. Untar it and follow the steps below.

Applying patches

$ cd compat-wireless-3.6-rc6-1
$ wget http://patches.aircrack-ng.org/mac80211.compat08082009.wl_frag+ack_v1.patch
$ patch -p1 < mac80211.compat08082009.wl_frag+ack_v1.patch
$ wget http://patches.aircrack-ng.org/channel-negative-one-maxim.patch
$ patch -p1 < channel-negative-one-maxim.patch

Choosing Driver

Choose the driver corresponding to the wireless device. In this TP Link WL721N is the USB wireless adapter card. It has an atheros chipset and the driver it uses is ath9k_htc ./scripts/driver-select ath9k_htc

Installation

$ make
$ sudo make install
$ sudo make wlunload
$ sudo modprobe ath9k_htc
A.2 Testing for frame injection support

The load test tool will work for sure if the following test is successful. Install Aircrack package from the url [17].

Creating a virtual wireless interface in monitor mode

Create a virtual wireless interface. Here, <interface name> is the name of the wireless adapter (typically wlan0 or eth0, etc). To find out the name of the interface use “ifconfig” command.

$ airmon-ng start <interface name>

Testing wireless association

After the above step, a new virtual wireless interface (typically named “mon0”) will be created. Run the following command to check whether it is possible to connect to the wireless access point using the virtual interface created.

$ aireplay-ng -1 6000 -o 1 -q 10 -e <SSID of Access point> -h <mac address of interface> mon0

After running the above command, if it says connection successful, then it means that the current wireless card can associate to wireless access point using its own mac address. Change the mac address to some random address and try the same command again. If its not able to connect, it means that the wireless adapter doesn’t support connection using random mac id. And load testing tool cannot work using this wireless card.
Appendix B

Graphs on All 3 Channels for Comparison

B.1 Goodput, TCP and Datalink adjustments

Figure B.1: Goodput data on Channel 1

Figure B.2: Goodput + TCP overheads on Channel 1
Figure B.3: Goodput + TCP overhead + Datalink overhead on Channel 1

Figure B.4: Goodput data on Channel 6

Figure B.5: Goodput + TCP overheads on Channel 6
Figure B.6: Goodput + TCP overhead + Datalink overhead on Channel 6

Figure B.7: Goodput data on Channel 11

Figure B.8: Goodput + TCP overheads on Channel 11
Figure B.9: Goodput + TCP overhead + Datalink overhead on Channel 11
Appendix C

Traffic Shaping Configurations

This section describes the configuration of tc command for traffic shaping. The article [18] gives so many configurations of the tc command suitable for various cases.

C.1 Experiment 1 Configuration

sudo tc qdisc del dev eth0 root
sudo tc qdisc add dev eth0 root handle 1: htb default 10
sudo tc class add dev eth0 parent 1: classid 1:1 htb rate 8mbit burst 15k
sudo tc class add dev eth0 parent 1:1 classid 1:10 htb rate 8mbit ceil 8mbit burst 15k
sudo tc qdisc add dev eth0 parent 1:10 handle 10: sfq perturb 10

C.2 Experiment 2 Configuration

sudo tc qdisc del dev eth0 root
sudo tc qdisc add dev eth0 root handle 1: htb default 10
sudo tc class add dev eth0 parent 1: classid 1:1 htb rate 16mbit burst 15k
sudo tc class add dev eth0 parent 1:1 classid 1:10 htb rate 8mbit ceil 8mbit burst 15k
sudo tc class add dev eth0 parent 1:10 handle 10: sfq perturb 10

C.3 Experiment 3 Configuration

sudo tc qdisc del dev eth0 root
sudo tc qdisc add dev eth0 root handle 1: htb default 20
sudo tc class add dev eth0 parent 1: classid 1:1 htb rate 16mbit burst 15k
sudo tc class add dev eth0 parent 1:1 classid 1:10 htb rate 16mbit ceil 16mbit burst 15k
sudo tc class add dev eth0 parent 1:10 handle 10: sfq perturb 10
sudo tc qdisc add dev eth0 parent 1:20 handle 20: sfq perturb 10
U32="tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32"
sudo $U32 match ip dport 22 0xffff flowid 10:1