Faculty of Engineering and Science
University of Agder

Master Thesis

Accurate Measurement and Visualization of Traffic Load in IEEE 802.11 WLANs

Harald Unander

Wang Wenjuan

May 2008

Abstract

Wireless Local Area Networks (WLANs) based on the IEEE 802.11 standard are getting very popular these days, due to their support for new multimedia services and VoIP, and because WLANs are appearing on more and more small devices. This causes a steadily increasing load and becomes a challenge to network administrators who want to make their WLANs run efficiently and to keep the users satisfied.

This Master thesis addresses the lack of efficient and affordable monitoring tools for those running WLANs, by developing an accurate method and for the visualization of traffic patterns and bandwidth utilization.

We have developed the Wlan Traffic Visualizer, which, based on captured log files or through live capture, gives a clear presentation of the traffic patterns and load situation of a WLAN, as well as the bandwidth utilization and potential extra throughput. We have implemented new methods for accurate free bandwidth estimation, and through a series of testing activities, our implementation has been verified in comparison with the standards and other references.

As an additional outcome, the Wlan Traffic Visualizer has proved to be useful for other purposes. Its detailed presentation of packet sequences and timing can be used for WLAN protocol illustration, as well as for the observation of WLAN equipment behaviour and for equipment benchmarking.

Preface

This thesis work concludes our two-year Master of Science program in Information and Communication Technology at the University of Agder, Faculty of Engineering and Science in Grimstad, Norway. The workload of this thesis is equal to 30 ECTS, and the project has been carried out from January to June 2008.

First of all, we would like to thank our principal supervisor Frank Li at the University of Agder, for bringing up the idea of this thesis, which supplements the EU FP6 IST ADHOCSYS project in which he is taking part, and for excellent supervision during the project.

The thesis has been carried out in cooperation with Ericsson Mobile Platforms (EMP), which has provided technical supervision and helped us with office space and test equipment. Our supervisors at EMP have been Øyvind Murberg and Kim Lilliestierna, and we would like to thank them for their assistance in uncovering the inner secrets of the 802.11 protocols, and for helping us keeping the right focus, in particular in our tool development effort.

Finally, we would like to thank DevoTeam and Per Gunnar Riber, as the employer of Harald, as well as EMP, for their positive attitude towards competence building and studies, and thereby making this master project possible.

Grimstad, 26 May 2008

_____	_____
Harald Unander	Wang Wenjuan

Table of contents

List of figures

List of tables

1 Introduction

1.1 Background

IEEE 802.11 Wireless Local Area Networks (WLANs) are becoming very popular and this trend seems to continue. In WLANs, stations associate with an access point to obtain network service and are also operative while they move across multiple access points. WLANs have become popular due to ease of installation and location freedom with the gaining popularity of laptops, PDAs and WLAN enabled smart phones. The benefit of WLANs is that they are convenient. Another benefit is that it is easy to deploy and requires little more than a single access point, and it is easily expandable.

With a great increase in the use of WLANs and new real-time services like VoIP, multimedia and video, most of the available bandwidth could often be used and the quality of service is degraded if a WLAN is overload. This will cause traffic congestion and the end users will be dissatisfied. Due to the lack of efficient tools and methods, it is difficult for network administrators and users to understand what is going on in their networks.

One possible solution to solve this problem is to measure traffic load by sniffing packets on the air going to and from an access point. This solution will be further addressed in this thesis.

1.2 Problem Statement

The aim of this thesis project is to develop an accurate method to measure and visualize free and busy bandwidth in IEEE 802.11 Wireless Local Area Networks. The project includes a pilot implementation of the method, and verification of the implemented method through testing.

The particular problems to be solved are:

How to collect data from the radio medium

In order to collect data, we need tools and equipment, and we also need a method to set up and configure these tools and equipment. In the setup, the location of the sniffers will greatly impact the quality or the logged data, or in other words, improperly placed sniffers will cause loss of packet in the logs. With a well planned interconnection of sniffers and other test equipment it will be possible to aggregate measurements from several sources and filter out noise.

How to calculate the bandwidth components

Parsing the deep protocol hierarchies in the capture files in order to extract time data and other information is one of the most complex parts of this thesis. In order to succeed we need to base our work on existing tools and open source solutions. In addition we need to consult the latest publications in the area to find the basis for accurate bandwidth calculations, and furthermore propose our calculation algorithm.

How to present the results to the users

It is not clear what data and presentation format that is beneficial for the user to get a clear understanding of situation of her/his WLAN. The collected data must be processed and presented, so that the user can easily interpret the different WLAN bandwidth components and other relevant data. There is a wide range of presentation formats to choose from. We must also consider both live monitoring and presentation of trends over different time periods.

How to verify the efficiency and precision

The quality and precision of the measurements in a simple WLAN configuration with one access point under optimal conditions can be verified. However, in a more complex situation with hidden terminals or interference problems it is not clear how the simple data collection will perform. There are also more complex WLAN configurations that we need to consider, like WMM with multiple multimedia streams and possibly more advanced WLAN standards providing higher bandwidth.

1.3 Motivation and Utility Value

This thesis work is triggered by an EU FP6 project called ADHOCSYS [1], and it is supposed to be a supplement to this EU project. At the same time, Ericsson Mobile Platforms is also very interested in this thesis project, as they are working with WLAN development for mobile phones.

Wireless LANs induce many technological possibilities that are not supported in wired networks, especially with mobility. Consequently there is an increasing number of WLAN users, and real-time services will soon take up a major part of the available bandwidth in IEEE 802.11 networks. Improving bandwidth utilization is therefore essential.

In the thesis, the task is to provide a way to track network traffic and find idle bandwidth. With our tool end-users will be aware of available resources for communication, and it will be easier for network administrators to observe their network's behaviour, find bottlenecks and improve its performance. In lack of such tools, traffic congestion, message loss, or bandwidth waste will often occur, and it is difficult to identify and improve the situation.

Our work is very challenging as there are not many studies on the measurement of available bandwidth, proving enough precision and a user-friendly interface with open source. With the completion of our task, our tools and measurement methods will be freely available for every network manager and end user interested in seeing how their WLAN network is behaving.

We hope to identify different WLAN usage patterns and configurations showing good or less good WLAN bandwidth utilisation or efficiency. Upon finding such patterns it should be possible to provide guidelines for efficient WLAN setup and usage. All in all we hope our efforts will lead to an increased number of satisfied WLAN users.

1.4 Method

To accomplish the objective of this thesis we first do a survey of existing methodology and the theoretical knowledge available for WLANs and WLAN measurements. According to the requirements given by the thesis definition, we have to select and develop the appropriate tools for data collection and analysis. Then, we use this knowledge to establish a tool chain and procedure for the collection and visualization of WLAN traffic and bandwidth usage. Finally, we run our solution through a series of trials in order to verify its efficiency and validity.

1.5 Delimitations and Assumptions

- Our method will support the following standards:
    - IEEE 802.11
    - IEEE 802.11a

  - IEEE 802.11b
  - IEEE 802.11g
- 802.11g will be our preferred WLAN standard for measurements.
- We choose Linux as our preferred sniffing platform, as it has open source drivers and tools supporting collection of WLAN MAC layer packets. Windows has no similar solution that is freely available. However, there are proprietary solutions in Windows that will be tried out, as a second choice.
- We assume there exist free tools that will help us collect the needed data to measure free bandwidth.
- We assume our experiment networks are sufficiently similar to the real wireless networks that consist of interference and a variety of traffic streams.
- We assume it is possible to test different scenarios that are sufficiently similar to scenarios occurring on real WLANs.
- We assume our measurements tools and techniques can be used in real life. Therefore these data and results supply a reference for companies and individuals. This means that they can become guidelines for the configuration of optimised parameter and WLAN entities with WMM.
- Our hypothesis is that flexible and inexpensive way of measure wireless networks will show improvement in performance in most scenarios.
- We realize that solving all the particular problems mentioned in Section 1.2 completely will require too much work for our thesis, and we will have to limit the scope. Our intention is to fully solve the data collection and the presentation problems, and to solve the measurement quality problem as precisely as possible.

1.6 Thesis Structure

This report is structured as follows:

Chapter 1 (this chapter) gives the problem statement including four particular problems and gives an introduction to the Master thesis.

Chapter 2 provides relevant information about IEEE 802.11 WLANs.

Chapter 3 presents existing theory, work and tools related to bandwidth measurements in WLANs.

Chapter 4 describes the method used and the core work of this thesis. Our solutions to the four particular problems are presented.

Chapter 5 describes the tool we have developed, the *WLAN Traffic Visualizer (WlanTV)*

Chapter 6 presents the experiments with results, in relation to theory and other references.

Chapter 7 discusses our results and give an overview of our findings.

Chapter 8 gives the conclusion of our thesis and points out directions for further studies.

2 A Survey of IEEE 802.11 WLANs

As a basis for our measurement methods development, we do a survey of WLANs and relevant IEEE 802.11 standards. We then we look at frame formats and frame exchange components on the physical and MAC layers and explain the hidden node and interference problems.

2.1 Wireless Local Area Networks

2.1.1 Introduction

In our thesis we define WLAN as being a wireless local area network of the 802.11 standards family.

> "A wireless LAN or WLAN is a wireless local area network, which is the linking of two or more computers without using wires. WLAN utilizes spread-spectrum or OFDM modulation technology based on radio waves to enable communication between devices in a limited area, also known as the basic service set. This gives users the mobility to move around within a broad coverage area and still be connected to the network." [31]

Wireless network is composed of three major components, the *stations*, the *wireless medium* and the *distribution system*.

2.1.1.1 Stations

All components that can connect into a wireless medium in a network are referred to as *stations* or *STA*s. All stations are equipped with wireless network interface cards (WNICs). Wireless stations fall into one of two categories: access points and clients:

> *Access points* or *AP*s are base stations for the wireless network. They transmit and receive radio signal for wireless enabled devices that they communicate with. They also perform the wireless-to-wired bridging function. That means this bridge allows the connection of devices on a wired network to a wireless network and also acts as the connection point to the wireless LAN.
>
> *Wireless clients* can be mobile devices such as laptops, personal digital assistants, IP phones, or fixed devices such as desktops and workstations that are equipped with a wireless network interface.

2.1.1.2 Wireless Medium

The function of wireless medium is to move frames from one station to another. There are different physical layers defined. Initially, radio frequency (RF) physical layer and infrared physical layer were standardized, but RF physical layer has been proven more popular.

2.1.1.3 Distribution System

A distribution system connects several access points in an extended service setup. The aim of a DS is to increase network coverage through roaming between access points. Thus it could forward frames to their destinations in another wireless network.

In WLANs, stations must associate with an access point to obtain network services and are also operative while they move. WLANs have become popular due to ease of installation and location

freedom with the gaining popularity of laptops. The benefit of the WLAN is its convenience. It allows users to access network resources from any location within its coverage area. Another benefit is that the WLAN is easy to deploy and requires little more than a single access point, and an increasing number of clients can be served without extra equipment. Additional clients would require additional wiring in a wired network.

The disadvantage is the lack of security related to the information exchange. The radio medium is available for everybody, which means that everyone is able to listen to wireless communication within the range, and therefore wireless communication is more vulnerable to abuse, intrusions and attacks than wired networks. Another problem is the relatively slow speed of most wireless networks. The speed on wireless network reaches at its best 108Mbit/s compared to the slowest modern wired networks at 100Mbit/s.

2.1.2 Network Types

The 802.11 wireless networks are built around the *basic service set* (BSS), which is a set of STAs communicating with each other. When a station is in the BSS area, it can communicate in two ways, as a member of an independent BSS (also referred to as IBSS) or an infrastructure BSS, as illustrated in Figure 1.

2.1.2.1 Independent Networks

The left part of Figure 1 shows the independent network architecture. When all of the STAs in the BSS are mobile STAs, and there is no relay connection to a wired network, the BSS is called an independent BSS (IBSS). Stations in an IBSS communicate directly with each other within a certain range. One common use is to create a short-lived network for specific purposes. Because the typical characteristic of the IBSS is its short lifetime, small range and specific purpose, it is also referred to as *ad hoc* networks.

2.1.2.2 Infrastructure Networks

The right part of Figure 1 shows an infrastructure BSS. The difference from the independent BSS is the use of an access point. An access points can be used for all communications between mobile nodes in its service area, and it can provide bridging to other service areas.

In the infrastructure network, stations must associate with an access point to obtain the access to network services. There is no theorethical limitation to the number of stations. However, the limited throughput of the WLAN is likely to limit the number of active stations in a infrastructure network.

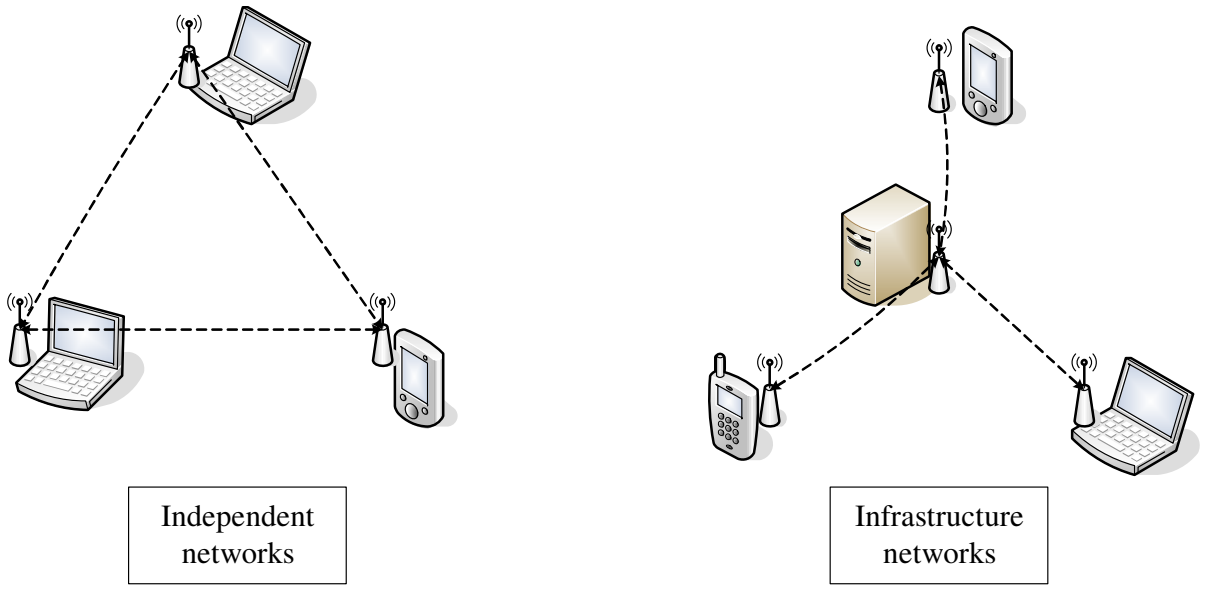


Figure 1 - Independent and infrastructure BSSs

2.1.3 Distributed Coordination Function (DCF)

CSMA/CA, Carrier Sense Multiple Access with Collision Avoidance, is the access mechanism that tries to avoid simultaneous access (collisions) by deferring access to the medium when it is busy. A series of coordination functions can control the access to the wireless medium. DCF, the distributed coordination function, is the basis of the standard CSMA/CA mechanism. If we require the free contention services, the point coordination function (PCF) can provide. PCF is built on top of the DCF. Networks can us the hybrid coordination function (HCF), the quality of service between DCF and PCF. PCF is only applicable in infrastructure networks, but HCF may be adopted in any network.

Because our research is based on DCF, we present this more in detail:

> "DCF requires a station wishing to transmit to listen for the channel status for a DIFS interval. If the channel is found busy during the DIFS interval, the station defers its transmission. In a network where a number of stations contend for the multi-access channel, if multiple stations sense the channel busy and defer their access, they will also virtually simultaneously find that the channel is released and then try to seize the channel. As a result, collisions may occur. In order to avoid such collisions, DCF also specifies random backoff, which forces a station to defer its access to the channel for an extra period.
>
> DCF also has an optional virtual carrier sense mechanism that exchanges short Request-to-send (RTS) and Clear-to-send (CTS) frames between source and destination stations during the intervals between the data frame transmissions." [30]

Another characteristic of DCF is its virtual carrier-sensing function. Virtual carrier sensing is provided by *the Network Allocation Vector (NAV)*. The NAV is a timer that when it is running indicates that medium is busy. A station sets the NAV as the duration field of its transmitted frames in order to show how long it expects to occupy the medium, and other stations will wait until the period has elapsed, before trying to contend for access.

More information about DCF can be found in [11].

2.2 WLAN Standards

2.2.1 IEEE 802.11

> "IEEE 802.11 is a set of standards for wireless local area network (WLAN) computer communication, developed by the IEEE LAN/MAN Standards Committee (IEEE 802) in the 5 GHz and 2.4 GHz public spectrum bands." [31]

Figure 2 shows the relationship between the 802 families and their position in the OSI model. IEEE 802 specification includes the lower two layers (physical layer and data link layer) of the seven-layer OSI network model. Actually the data link layer in IEEE 802 is split into two sub-layers named logical link control (LLC) and medium access control (MAC). The MAC controls how to access the medium and send data, but the details of transmission and reception is decided by the physical layer.

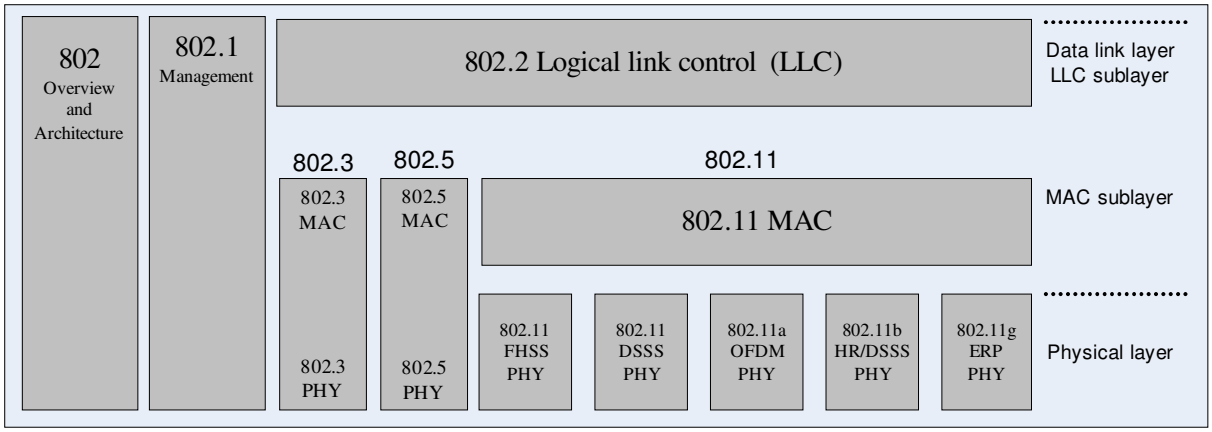


Figure 2 - The IEEE 802 family

From the Figure 2, basic 802.11 specification holds the lower two layers at the same time. That means we could use a series of 802.11 specifications to define wireless networks independent of other standards.

Although the 802.11 has a maximum MSDU size of 2304, the maximum practical frame length is limited by the MTU size set at the 802.2 LLC layer, and this size is normally set to 1492. Increasing the application packet size to above 1492 will cause IP fragmentation and 802.11 will only have packets shorter than 1492.

2.2.2 802.11a

> "The 802.11a, an amendment to the original standard, was ratified in 1999. it uses the same core protocol as the original standard, operates in operates in 5 GHz band, and uses a 52-subcarrier orthogonal frequency-division multiplexing (OFDM) with a maximum raw data rate of 54 Mbit/s. The data rate can be reduced to 48, 36, 24, 18, 12, 9 then 6 Mbit/s if required. 802.11a is not interoperable with 802.11b as they operate on separate bands, except if using equipment that has a dual band capability. Nearly all enterprise class Access Points have dual band capability." [32]

Orthogonal frequency division multiplexing (OFDM) is used to chop a large frequency channel into a number of sub channels. The sub channels are then used in parallel for higher throughput. That is why OFDM has fundamental propagation advantages when in a high multipath environment, such as indoor office and high building which counteract the transmission of single.

2.2.3 802.11b

"802.11b has a maximum raw data rate of 11 Mbit/s and uses the same CSMA/CA media access method defined in the original standard. Due to the CSMA/CA protocol overhead, in practice the maximum 802.11b throughput that an application can achieve is about 5.9 Mbit/s using TCP and 7.1 Mbit/s using UDP.

802.11b products appeared on the market in early 2000, since 802.11b is a direct extension of the DSSS (Direct-sequence spread spectrum) modulation technique defined in the original standard. Technically, the 802.11b standard uses Complementary code keying (CCK) as its modulation technique. The dramatic increase in throughput of 802.11b (compared to the original standard) along with simultaneous substantial price reductions led to the rapid acceptance of 802.11b as the definitive wireless LAN technology." [33]

There are two kinds of modes supported by 802.11b devices using different modulation technologies, respectively DSSS and HR-DSSS. For DSSS PHY, the supported data rates have 1 Mbps and 2 Mbps. However, a PHY with data rates of 5.5 Mbps and 11 Mbps was specified in 1999. The old data rates combined with the new data rates to generate a new single interface called HR-DSSS. They both operate in the 2.4 GHz band.

2.2.4 802.11g

"802.11g, the third modulation standard for Wireless LAN, is not a revolutionary specification. In fact, it is clear that it uses too much of the existing work done. There are only slight modifications in physical layers. It works in the 2.4 GHz band (like 802.11b) but operates at a maximum raw data rate of 54 Mbit/s, or lower data rate. In an 11g network, however, the presence of a legacy 802.11b participant will significantly reduce the speed of the overall 802.11g network.

The modulation scheme used in 802.11g is orthogonal frequency-division multiplexing (OFDM) copied from 802.11a with data rates of 6, 9, 12, 18, 24, 36, 48, and 54 Mbit/s, and reverts to CCK (like the 802.11b standard) for 5.5 and 11 Mbit/s and DBPSK/DQPSK+DSSS for 1 and 2 Mbit/s. Even though 802.11g operates in the same frequency band as 802.11b, it can achieve higher data rates because of its heritage to 802.11a. Totally 802.11g contains the following components, such as ERP-DSSS, ERP-OFDM, ERP-PBCC, DSSS-OFDM and CCK-OFDM, but ERP-OFDM is the major mode of 802.11g." [35]

2.2.5 802.11e

"IEEE 802.11e-2005 or 802.11e is an approved amendment to the IEEE 802.11 standard that defines a set of Quality of Service enhancements for wireless LAN applications through modifications to the Media Access Control (MAC) layer. The standard is considered of critical importance for delay-sensitive applications, such as Voice over Wireless IP and Streaming Multimedia. The amendment has been incorporated into the published IEEE 802.11-2007 standard." [34]

2.2.6 Wi-Fi

"Wi-Fi is a wireless technology brand owned by the Wi-Fi Alliance intended to improve the interoperability of wireless local area network products based on the IEEE 802.11 standards. Common applications for Wi-Fi include Internet and VoIP phone access, gaming, and network connectivity for consumer electronics such as televisions, DVD players, and digital cameras.

The Wi-Fi Alliance is a consortium of separate and independent companies agreeing to a set of common interoperable products based on the family of IEEE 802.11 standards.

The Wi-Fi Alliance certifies products via a set of established test procedures to establish interoperability. Those manufacturers that are members of Wi-Fi Alliance whose products pass these interoperability tests can mark their products and product packaging with the Wi-Fi logo." [38]

2.2.7 WMM

"Wi-Fi Multimedia (WMM) is a Wi-Fi Alliance interoperability certification, based on the IEEE 802.11e draft standard. It provides basic Quality of service (QoS) features to IEEE 802.11 networks. WMM prioritises traffic according to four AC (Access Categories) - voice, video, best effort, and background. However, it does not provide guaranteed throughput. It is suitable for simple applications that require QoS, such as Voice over IP (VoIP) on Wi-Fi phones." [13] [39]

2.3 Frame Formats

The frame format is important for the understanding of the IEEE 802.11 standard. There are numerous differences between the frame structures of wired networks and wireless networks, and it is essential to secure the operation on the unsafe and unprotected broadcast channel. Figure 3 shows the details of Ethernet 802.3 frame format that is used for wired networks, and in the following part we will present the structure of the 802.11 frames to show the main different frame formats on MAC layers between wired wireless networks.

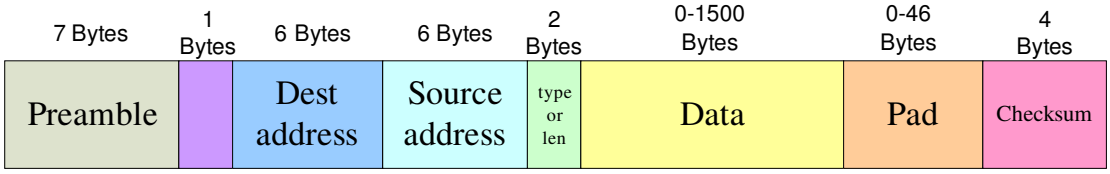


Figure 3 – Ethernet 802.3 MAC frame

2.3.1 802.11 MAC Header

As we can see from Figure 4, the MAC layer adopt several unique features to meet the challenges of the wireless data link, including four address fields and FCS. In addition, frame control, duration and sequence control fields are all identified. All the fields are transmitted form left to right. The MAC frame follows the physical header. The next part will give a description about how to enable successful transmission on the physical layer.

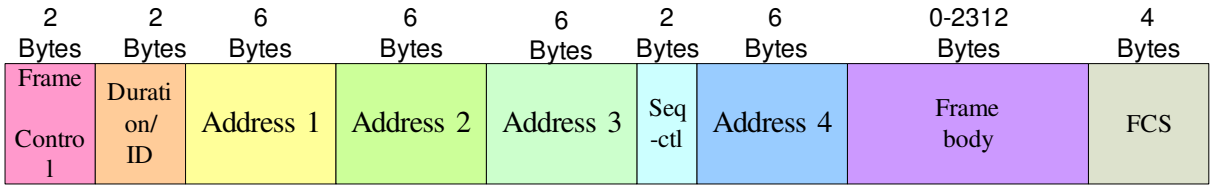


Figure 4 - Generic 802.11 MAC frame

Although the frame formats are not the same for all frame types, we won't consider every difference in here, but pay more attention to general structure according to Figure 4. Frame control with two bytes consists of several subfields of which the components are protocol version, subtype, type, more fragments, retry and so on. Two of the most interesting fields are the *type* and *protected frame* information. The *type* and *subtype* fields denote three kinds of frame, respectively data frames, control

frames and management frames. In total there are 36 different frames defined. More frame details will be discussed later. The *protected* frame bit is used for enabling security for a transmitted frame.

The *duration* field represents the time that the station expects to reserve the channel busy for itself. Any station could decode the headers of all frames that they receive in order to update its NAV.

There are the huge differences about address fields between wired and wireless frame structures. Depending on frame type, the four addresses can have different purposes. Generally, the first address indicates the receiver of frame, the second one is the transmitter address, the third one is for filtering by access point and distributed system, and the fourth address field is only used by wireless bridges.

In the case of IBSS, no access points or distribution system are used. Then the transmitter is the source, and the receiver is the destination, and the BSSID is created randomly. The BSSID is written into the 3rd address.

Between the third and fourth address, we have the *sequence control* field containing fragment number and sequence number. The sequence number is the frame counter that is increased for every transmitted frame. In case frames are bigger than the maximum fragment size it will be split into fragments and the *fragment number* counts fragments within a frame.

FCS is the last field and it is the Frame Control Sequence or Cyclic Redundancy Check. In wired and wireless network, the FCS field is used for confirming a flawless frame reception. All the frames with a bad FCS are discarded and have to be retransmitted. The frames with a good FCS are passed on to the protocol stack.

2.3.2 Physical Header (PLCP)

Before introducing the timing on the MAC layer we should look at the physical headers that are appended in front of MAC protocol headers. The physical layer convergence procedure (PLCP) helps the frame transfer at the physical layer, and it consist of two components. The first is the PLCP preamble with the preamble synchronization field and start frame delimiter field. The sync field is used to keep the receiver and sender synchronized so that the frame is decoded successfully, and the end of the preamble, the SFD signal, marks the beginning of the frame. These two fields are retained when channel mode changes. After the PLCP preamble follows the PLCP header.

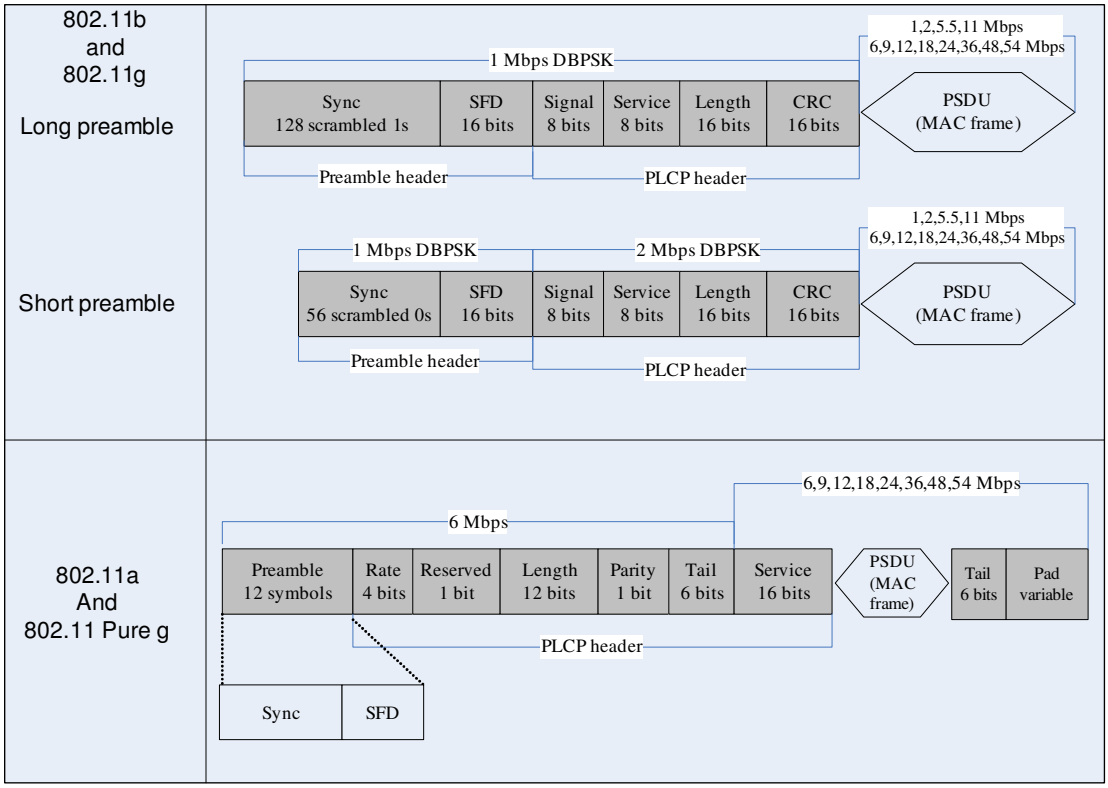


Figure 5 - PLCP frame in 802.11

In Figure 5, we can see that the structure of PLCP header is more complex than that of preamble header, for it varies according to the different standards or channel modes. In the first case, for 802.11b and the combined 11g/11b, the PLCP header comprises four fields. The receiver identifies the transmission rate of the encapsulated MAC frame from the signal field, and it gets the length of the MAC payload from the length field. To protect the header against corruption on the radio link, the senders calculate a CRC checksum and the receivers verify the CRC to make sure that the header is not damaged during transmission. In the second case, for 802.11a and 802.11g, as shown in Figure 5, the components of PLCP header are composed of five fields. However, we just pay our attention to the *service* and the *trailer* fields because they are different from the first case. The 16-bit *service* field is transmitted at the data rate of the MAC frame, unlike the other components of the PLCP header. The trailer field includes the *tail* and the *pad* bytes.

In addition, we also notice that transmission data rates are different for the preamble header and the PLCP header. For 802.11b and combined 11g/11b networks, preamble headers are transmitted at 1Mbps, and PLCP headers are transmitted at 1Mbps (long preamble) or 2Mbps (short preamble). 6Mbps is used for 802.11a and 802.11g networks.

When calculating the duration of a frame, the physical headers are appended and have to be added to our calculation. Table 1 represents all parameters' values in different channel modes.

Standard	*Mode*	*Data Rates (Mbps)*	*Preamble*	*Preamble Length (us)*	*PLCP Header (us)*
802.11	FHSS	1,2		96	32
802.11a	OFDM	6,9,12,18,24,36, 48,54		16	4
802.11b	HR-DSSS	1,2,5.5,11	Long	144	48
		1,2,5.5,11	Short	72	24
	DSSS	1		144	48
802.11g	ERP-OFDM	6,9,12,18,24,36, 48,54		16	4
	DSSS-OFDM	6,9,12,18,24,36, 48,54	Long	144	48
		6,9,12,18,24,36, 48,54	Short	72	24
	ERP-PBCC	33,22	Long	144	48
		33,22	Short	72	24
	CCK-OFDM	6,9,12,18,24,36, 48,54	Long	144	48
		6,9,12,18,24,36, 48,54	Short	72	24

Table 1 - The components of the physical header of IEEE 802.11a/b/g

2.4 Frame Exchange Components

2.4.1 Frame Types (Management, Data and Control Frame)

In this sub-chapter, we will introduce the three main frame types, respectively the *data frame*, the *control frame* and the *management frame.*

Let's focus on the data frame firstly. The data frame is the primary frame type, hauling data from station to station. All other frames are there to assist data frame in reaching the terminal reliably. The data frame types can be categorized to eight different subtypes, as shown in Table 2, but we don't explain every data frame in here.

Control frames are always used in conjunction with data frame to perform acquisition operations and positive acknowledgment of received data. Table 2 shows the different control frames. From the frame format, the distinguishing feature of the control frame is the address field. There are four addresses in the basic 802.11 frame format, but for the control frames only two are used to identify sender and receiver.

Management frames perform supervisory functions. They are used to join and leave wireless networks and move associations from one access point to another. Most names of management frames are related to the function of frames, so it is easy to understand what is the exact usage.

Data frame	*Control frame*	*Management frame*
Data	RTS	Beacon
Data+CF-Ack	CTS	Probe_Req
Data+CF-Poll	Ack	Probe_Res
Data+CF-Ack+CF-Poll	PS-Poll	Association_Req
Null	CF_End	Reassociation_Req
CF-Ack	CF_End_Ack	Association_Res
CF-Poll		Disassociation
CF-Ack+CF-Poll		Disassociation_Res
		Authentication
		Deauthentication
		Action frame
		ATIM

Table 2 - Frame types in IEEE 802.11

2.4.2 SIFS, DIFS and EIFS

DCF was described in Section 2.1.3 and we can move on to describe the inter frame spaces or *IFS* of the frame exchange of the 802.11. The interframe spacing plays a large role in coordinating access to the transmission medium, and we will take closer look at SIFS, DIFS and EIFS.

The short interframe space (SIFS) is used for the highest priority transmissions of an ACK frame, and the CTS/RTS frame. When stations have seized the medium and need to keep it for the duration of frame exchange to be performed, SIFS is used between frames to prevent other stations from using the medium.

The second is the DCF interframe space (DIFS). As already mentioned a new transmitted frame is dependent on waiting for a DIFS before any transaction can begin. The DIFS is dependent on the SIFS and slot time. The values for the SIFS, DIFS and slot times are given in Table 3, and we can notice that the values of SIFS and DIFS are quite small.

$$DIFS = 2 \times Slot\_time + SIFS \quad (1)$$

The extended inter-frame space (EIFS) is not a fixed interval. It is used only when there is an error in a frame transmission. If a corrupted frame is received, the station sending the frame needs to wait an EIFS and not a DIFS before sending. The formula is followed as:

$$EIFS = SIFS + DIFS + ACKT_x Time \quad (2)$$

2.4.3 Backoff Time

After a frame transaction has finished and DIFS has elapsed, stations must wait for a period, or *defer*, until it can start transmission. This period of time is called the contention window or backoff window. This window is divided into slots, and the slot time is given by the channel mode. When a station attempts to access the medium, it picks a random slot number between zero and its current contention window (CW). The station that picks the lowest slot number will start transmitting first and it will gain access to the medium. The contention window size is doubled with every retransmitted frame, until the maximum size (CW_{max}) is reached. Once a frame is successfully transmitted, the contention window is set to its minimum value (CW_{min}) for the next transaction. This way, all stations are given a fair share of the available bandwidth, and stations close to the coverage limit of the access point are not allowed to steal all the bandwidth for retransmission attempts.

	802.11	*802.11a*	*802.11b*	*802.11g*
Slot time (μs)	50	9	20/9	20/9
CW_{min}	15	15	31	15
CW_{max}	1023	1023	1023	1023
SIFS (μs)	28	16	10	10
DIFS (μs)	128	34	50/28	50/28

Table 3 - Intervals (IFS) information for IEEE 802.11 a/b/g

2.5 The Hidden Node Problem

Unlike in a wired network, the signal in a wireless network is not only dependent on the location and distance to the peer, but it varies over time as stations enter and leave the network, and due to interference and collisions with neighbour networks. In this part, we will discuss the hidden node problem.

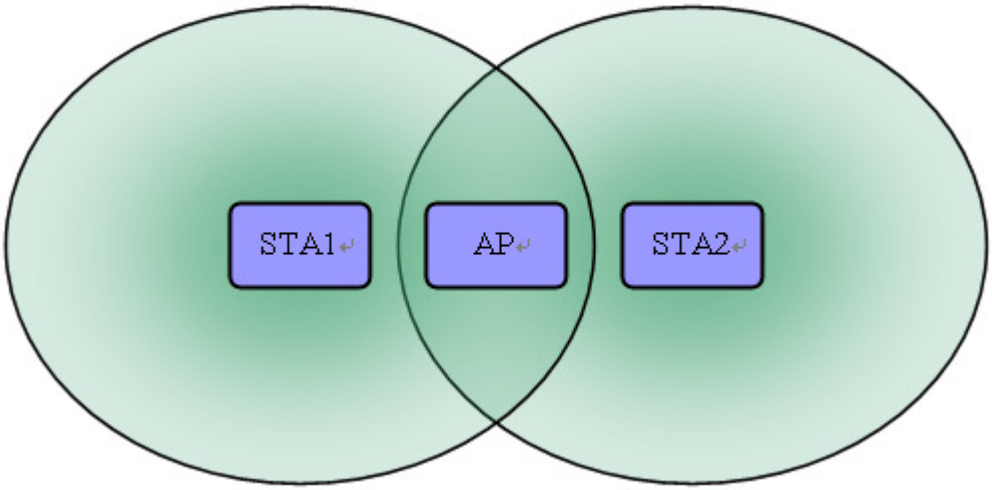


Figure 6 - The hidden node problem

In the Figure 6, STA1 can communicate with the AP, and STA2 can also talk with the AP. But STA1 cannot hear STA2 and STA2 cannot hear STA1. From the perspective of STA1, STA2 is the hidden node, and vice versa. This means that STA1 and STA2 can start transmission at the same time, without noticing, there is a collision, and both frames are corrupted and cannot be received at the AP. As 802.11 has grown up, the hidden station problem has been avoided by using RTS/CTS frame to reserve the radio link for transmission.

However the practical situation is more complex. In larger environments, the coverage is dense enough so that the stations could be able to connect with two or more access points at the same time. Imaging that there are three access points at the same channel, three wireless networks have overlapped area. The client in this area can transfer and receive frames with the other clients distributed over three different wireless networks. The hidden node problem is then more difficult to solve than the simple case illustrated in the above figure.

2.6 Interference and Obstructions

When enjoying the convenience and happiness without wires in WLAN, we're also aware of unavoidable enemy: interference. If having experienced interference, we know it is not easy to avoid. To solve the interference problem, the first key point is to know what the interference is. There are two types of interference mentioned here. The first interference comes from other radio cells on the same channel frequency. In order to decrease this interference, we could rearrange the cells of a certain channel frequency so that there is less overlap. The second one is the resulting from neighbouring WLANs on different channels.

In practice, before selecting a WLAN on our computer, or measure the strong strength of the WLAN, some parameters are measured and can be depending on the equipment, from [27]:

- Signal strength: Measured in decibels compared to one milliwatt (or dBm), the signal strength is sometimes referred to as signal level. The higher this number is, the better chance you have for a full-speed connection between your access point and your PC.
- Noise level: Ideally, you want the noise level (also measured in dBm) to be as low as possible. Cordless phones and microwaves are common culprits for increasing the noise level.
- Signal-to-noise ratio (SNR): This is the most telling of the numbers because it compares the strength of the signal with the noise that is interfering. SNR is measured in decibels (dB), and a higher number is good news.

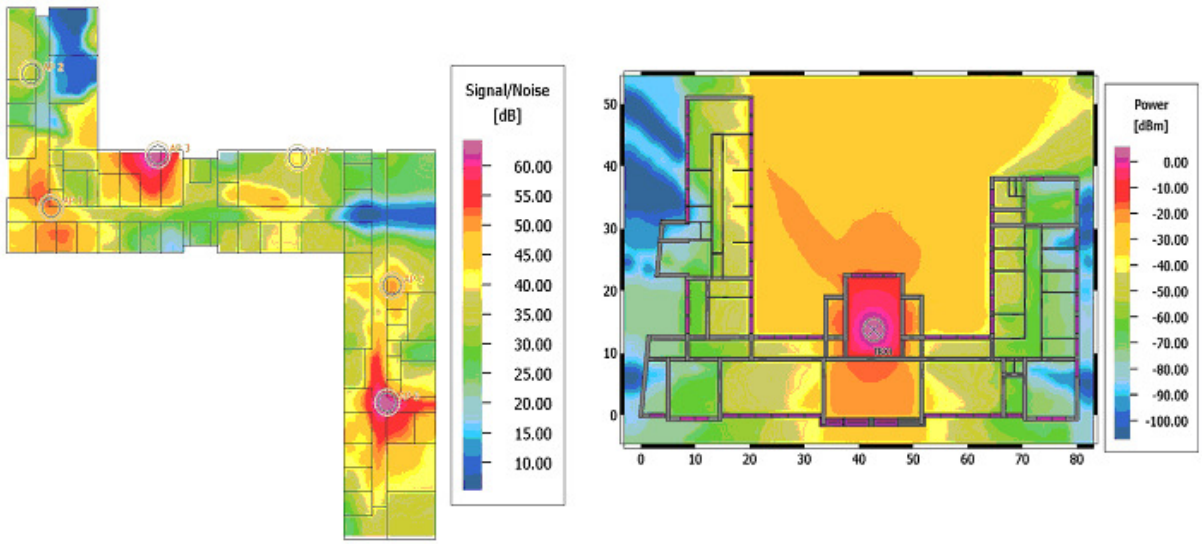


Figure 7 – A simulation showing the field strength in a room in 802.11 wireless networks from [28]

Another factor with impact on the signal strength is the obstructions. Figure 7 shows highly accurate capacity and data throughput based on density definitions and interference analysis. The commercial 802.11 cards available today differ greatly in emitted signal strength for symmetrical transmission.

We can see that is preferred to choose a location close to the access point. When the configuration of a new wireless network, the person being full of experience can avoid potential interference and frustration.

3 Bandwidth Measurements in WLANs

The primary objective of this chapter is bandwidth measurements on the MAC layer. Continuing the survey, we learn the principle of the theoretical maximum throughput and two different ways of bandwidth estimation: passive sniffing and active probing. After comparing the two methods and practical experience, we chose the passive sniffing for our thesis, so this approach is described in depth in order to give a smooth the progression to Section 4.2, in which our bandwidth calculations are presented. The last section of this chapter goes through previous related work and tools.

3.1 Throughput and Bandwidth

The meaning of the terms *throughput* and *bandwidth* have different meaning dependent on the context in which they are used. From Wikipedia, we found the following definitions:

> "*Digital bandwidth* or just *bandwidth* often refers to a data rate measured in bits/s. The limit to the data rate of a physical communication link is related to its bandwidth in hertz, sometimes denoted analogue bandwidth in computer networking literature." [29]

> "*Throughput* is the average rate of successful message delivery over a communication channel. The throughput is usually measured in bits per second (bit/s or bps), and sometimes in data packets per second or data packets per timeslot." [37]

In this thesis we will use the term throughput to denote the practical data rate or user data rate that can be obtained over the WLAN link, while the term bandwidth is used to denote a part or percentage of the total capacity that is available on a WLAN channel.

Every 802.11 standard has different maximum data rates, such as 54 Mbit/s for 802.11a, 11 Mbit/s for 802.11b and 54 Mbit/s for 802.11g. These are the data rates on the physical layer without overhead like the preamble and PLCP header, and they are far above data rates that can be obtained for user data.

The maximum theorethical throughput for user data is described in Section 3.1.

3.2 Theoretical Maximum Throughput (TMT)

> "With the development of IEEE 802.11 networks, new requirements like high data rate and throughput are expected by users, so systems designers and researchers in communication fields are always interested in knowing the how is the performance of a network system. They often select the most effective architecture or design constraints for a network, which can match with its final performance. Thus in most case, what the user or designer think about is the benchmark of what a network is capable of. Maximum throughput is one of the important benchmark. This value is a strict barrier that cannot be overcome by any means while remaining standard-compliant." [37]

There is a paper that describes the exact calculation of the theoretical maximum throughput (TMT) for 802.11 networks. TMT is defined "as the average maximum number of messages successfully delivered per unit time". Although it is possible to count the throughput at the different layers, this is considering the MAC layer. The following part introduces TMT as described in [6].

Before beginning to talk about the calculation formula, we should mention the conditions for the TMT calculation. The first assumption is that the network is perfect without any error and losses due to collisions. Secondly, there are sufficient packets waiting by the sender, and no packets loss due to buffer overflow on the receiving nodes. At last, fragmentation and management frames such as beacon and association frames are not considered. However, we know these assumptions are not valid in a real

network. That is means the values counted in the perfect environment are lower than in any real environment.

The TMT calculation is classified by different spectrum technologies, MAC schemes, basic data rates and packet sizes. Considering the MAC schemes, there are two sets, the CSMA/CA and RTS/CTS. Under every MAC scheme, calculations are divided to four different spectrums, FHSS, DSSS, HR-DSSS and OFDM. Table 4 shows all the schemes.

The calculations in paper [6] consider the total delay time per MSDU. The basic formula is:

$$TMT = \frac{MSDUsize}{delayPerMSDU} \qquad (1)$$

Through analysing the overhead at different sub layers and different time transmission for CSMA/CA and RTS/CTS, the total delay per MSDU is summed up, then the author deduced a simple formula with one variable instead of the total delay formula. Thus TMT is related to the number of bits in MSDU and two constants, a and b.

$$TMT(x) = \frac{8x}{ax+b} * 10^6 bps \qquad (2)$$

Table 4 gives the parameters for all schemes.

Scheme	*Data Rate*	***a***	***b***
CSMA/CA			
FHSS	1 Mbps	8.25	1179.5
	2 Mbps	4.125	1039.25
DSSS	1 Mbps	8	1138
	2 Mbps	4	1002
HR-DSSS	5.5 Mbps	1.45455	915.45
	11 Mbps	0.72727	890.73
OFDM	6 Mbps	1.33333	223.5
	12 Mbps	0.66667	187
	24 Mbps	0.33333	170.75
	54 Mbps	0.14815	159.94
RTS/CTS			
FHSS	1 Mbps	8.25	1763.5
	2 Mbps	4.125	1623.25
DSSS	1 Mbps	8	1814
	2 Mbps	4	1678
HR-DSSS	5.5 Mbps	1.45455	1591.45
	11 Mbps	0.72727	1566.73
OFDM	6 Mbps	1.33333	337.5
	12 Mbps	0.66667	273
	24 Mbps	0.33333	244.75
	54 Mbps	0.14815	225.94

Table 4 - TMT parameters for different MAC schemes and spread spectrum technologies, from [6]

According to above description, four TMT curves are represented in [6].

Figure 8 shows the curves that are relevant for our thesis. We will use the TMT in the testing chapter to compare with our testing results. Our measured bandwidth should be less than the TMT, for the tested data rates and packet lengths.

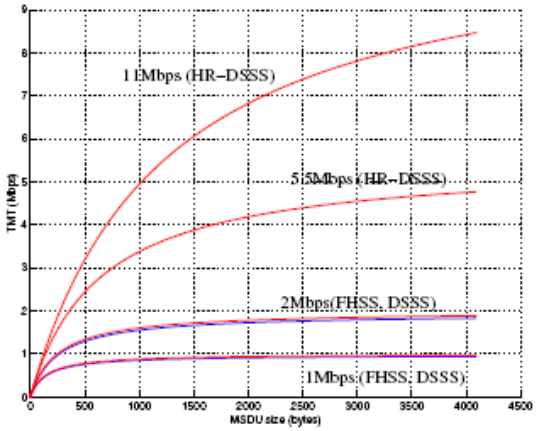


Figure 4. TMT curve for CSMA/CA - FHSS, DSSS, HR-DSSS

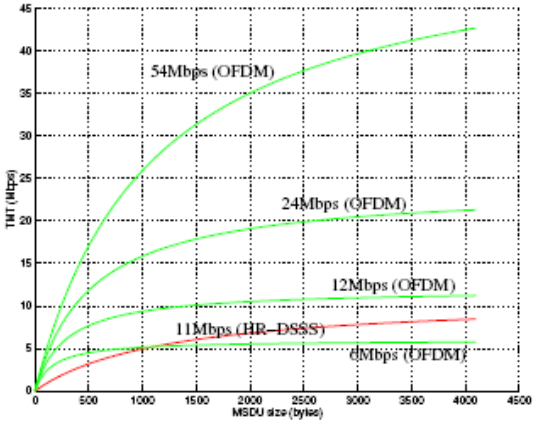


Figure 6. TMT curve for CSMA/CA - 11 Mbps HR-DSSS, OFDM

Figure 8 - TMT per frame size for different 802.11 modes from [6]

3.3 WLAN Measurements

There are two intuitive ways of measuring available bandwidth, through *passive sniffing* and *active probing*. The two methods are presented below.

3.3.1.1 Measurement Approach – Passive Sniffing

In order to avoid the affecting the network behaviour, passive sniffing can used to find used bandwidth by listening to traffic on a selected channel. When comparing with active probing, the difference is that we don't need to do anything except setting WNIC into monitor mode, and then begin listening. As discussed we have the virtual carrier sensing mechanism represented by the NAV to tell how long the medium will be busy or not. By using the NAV value in addition to the time data from the logged packets, we can find the total busy time and other bandwidth components.

This is the passive approach that is that can be accomplished with standard WNIC cards.

We define passive sniffing as the logging of WLAN MAC layer packets on the air for a single WLAN channel. Then, by taking the different information elements in the logged packets it is possible to calculate the bandwidth components. This has already been described in [8]. More details will be illustrated in the next section.

3.3.1.2 Measurement Approach – Active Probing

As the passive sniffing has the risk of underestimating the channel usage in case of collisions and lost, active sniffing is brought out as another new approach. It is described in paper [4]. In this paper, the main idea is to observe the channel occupancy status by sending an irregular series of probe packets and measure the response times. This method will not be used in our thesis, so we don't discuss this approach in detail. The following sections we pay attention to illustrate our passive sniffing approach.

3.4 Passive Sniffing Proposal

We chose the passive sniffing method as our free bandwidth estimation method. The theoretical decomposition of bandwidth is the basis of our bandwidth measurement and is given first. Then, after illustrating the transaction principle, we to discuss how to count the busy, access and free bandwidth.

3.4.1 Bandwidth Components

The definitions of MAC bandwidth components are flexible, but we decided to consult the definitions in paper [8]. Figure 9 represents a compact and intuitive description of MAC bandwidth components that is particularly suited to radio resource management schemes. :

> "A load bandwidth (BW_{load}) that is associated with the transmission of the data frames, an access bandwidth (BW_{access}) associated with the contention mechanism (whereby a station wins access to the wireless medium) and a free bandwidth (BW_{free}) that is associated with the QoS." [8]

If these three components are summed together, the total value is equal to entire traffic load.

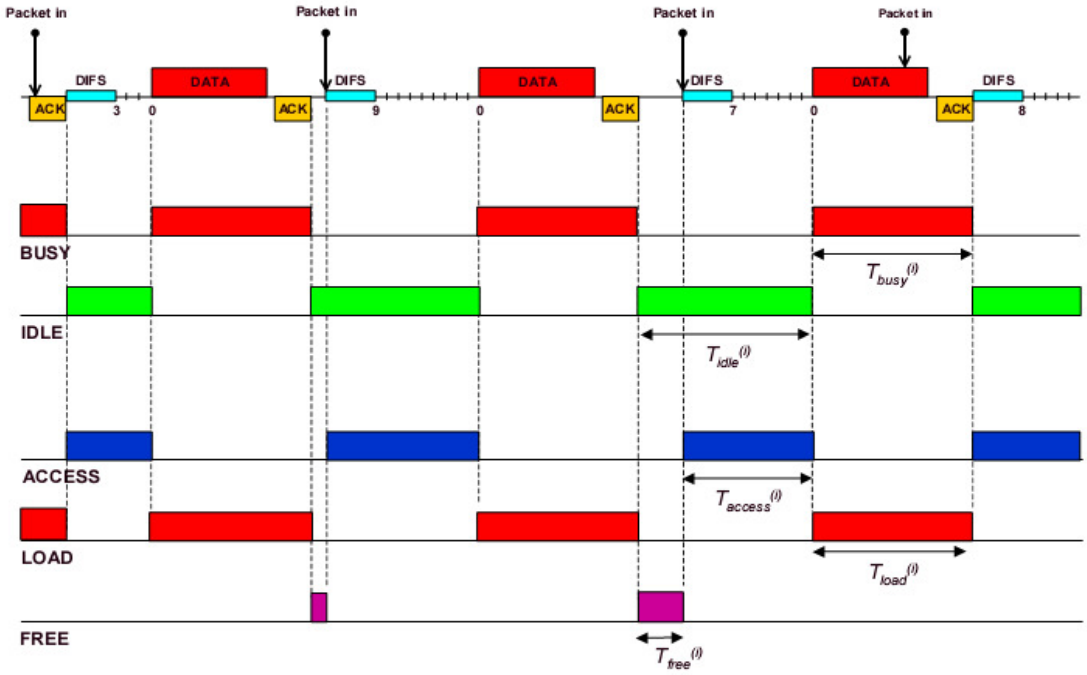


Figure 9 - Bandwidth components, from [8]

Two more components, BW_{busy} and $BW_{idle,}$ are also defined. At first BW_{busy} illustrates the interval that the medium is busy containing the frames transmission and their related acknowledgments. The remaining time is BW_{idle}. The stations use the idle time to defer (DIFS or EIFS) and for contention (backoff time). Thus the idle time consists of the free time and the access time.

We have five bandwidth components in total. Even if we know the five components, it is still not trivial to find the bandwidth components. Keep in mind that the busy time for a station is not equal to the busy time for the network. For a station, the time waiting for the medium to be free is considered busy time. However, for a network, or when sniffing on a network, the medium only appears busy when there is a transmission going on.

3.4.2 Full-scale Analysis of Transactions

A sequence of link-layer frames used to deliver a single management or data frame represents *a frame transmission transaction*. The paper [5] gives an introduction to the 802.11 transactions. This transaction is essential for our bandwidth component calculations and theory form this paper is presented here.

Note that only one transaction can occur at any point of time. In every transaction, DIFS and backoff time represent the access time, and the busy time of a transaction contains all frame time and interframe between frames in one transaction. Free time, however, is the available time between two transactions, not including the access time. Although it may look straightforward, an accurate estimation of the true channel time is difficult for a number of reasons:

Firstly, the backoff intervals are not fixed. Every station selects the contention window size based on different factors, such as retransmission and channel type, and depending on the current value of its backoff timer its contention time will vary. Secondly, there is a lot of interference in IEEE 802.11 WLANs because of the use of the unlicensed ISM band, and this means that the sniffer tool may not capture all frames on the radio channel. Lost frames and corrupted frames cannot be monitored, and the sniffer cannot filter out the frames coming from other WLANs. At last, the hidden node problem has an influence on the accuracy of our measurements. Although these limitations cannot be neglected, we provide the theoretical formulas to have exact presentation.

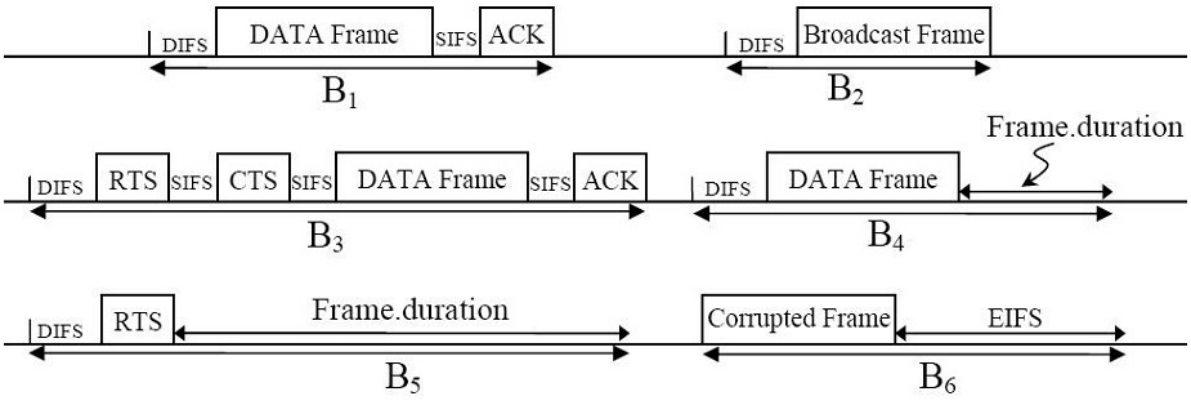


Figure 10 - Transaction cases from [5]

In Figure 10 the first three transactions show complete transactions. B_1 and B_3 are the unicast transactions with and without RTS/CTS, and B_2 is an example of a broadcast transaction. B_4 and B_5 show no-reply transactions, while B_6, the last one, is the corrupted transaction.

- When a transaction is complete and successful to be captured by sniffer tools, equation 1 is used to calculate the busy time.

$$T_{busy} = \sum_{k=1}^{N} Frame_k + (N-1) * SIFS \qquad (1)$$

N is the number of frames in the transaction, and $Frame_k$ represents the k th frame in the transmission transaction. For example, the data transmission with RTS/CTS, N=4 and $\sum_{k=1}^{N} Frame_k = Frame_{RTS} + Frame_{CTS} + Frame_{data} + Frame_{ACK}$.

- In the second transaction case the sender receives no response from receiver. In CSMA/CS, a data frame always reserve the channel for the following frame in the same transaction, all the other stations consider the medium to be busy during the time called NAV time.

$$T_{busy} = Frame + Frame * T_{NAV} \quad (2)$$

- The last equation is for the corrupted transaction. EIFS can be calculated by the Equation in Section 2.5.2.

$$T_{busy} = Frame + EIFS \quad (3)$$

- The access time is given by this formula:

$$T_{access} = DIFS + T_{backoff} \quad (4)$$

We give an example to illustrate the method of a total transmission transaction. Assuming a normal transaction with RTS/CTS carrying a data frame with the length of 200bits at 24Mbps and this transmission operates in the 802.11g mode. The first step is to count the frame busy time according to the equations in Table 5. Please note the use of the ceiling function and units of frame length and data rate in the formulas.

$$T_{frameBusy} = 26 + \lceil (22 + L)/4 * R \rceil * 4 = (26 + \lceil (22 + 200)/(4 * 24) \rceil * 4)\mu s$$

In addition, RTS (160 bits), CTS (112 bits) and ACK (112 bits) are sent at 2Mbps (DSSS PHY). Total busy time in the total transaction can be calculated like:

$$T_{RTS} = 192\mu s + L/R = 192\mu s + 160/2\mu s = 272\mu s$$

$$T_{CTS} = T_{ACK} = 192\mu s + L/R = 192\mu s + 112/2\mu s = 248\mu s$$

$$T_{busy} = T_{RTS} + T_{CTS} + T_{frameBusy} + T_{ACK} + 3 * SIFS = 272\mu s + 248\mu s + 36\mu s + 248\mu s + 3 * 10\mu s = 834\mu s$$

$$T_{access} = T_{DIFS} + T_{backoff} = 28\mu s + 68\mu s = 96\mu s$$

Thus this is the whole process of counting busy time and access time.

Standard	*Mode*	*Data Rates (Mbps)*	*Preamble*	*Frame time (us)*
802.11	FHSS	1,2		$128+\lceil L*33/32\rceil/R$
802.11a	OFDM	6,9,12,18,24,36, 48,54		$20+\lceil(22+L)/(4*R)\rceil*4$
802.11b	HR-DSSS	1,2,5.5,11	Long	$192+L/R$
		1,2,5.5,11	Short	$96+L/R$
	DSSS	1		$192+L/R$
802.11g	ERP-OFDM	6,9,12,18,24,36, 48,54		$26+\lceil(22+L)/(4*R)\rceil*4$
	DSSS-OFDM	6,9,12,18,24,36, 48,54	Long	$210+\lceil(22+L)/(4*R)\rceil*4$
		6,9,12,18,24,36, 48,54	Short	$114+\lceil(22+L)/(4*R)\rceil*4$
	ERP-PBCC	33	Long	$193+\lceil(8+L)/R\rceil$
		22		$192+\lceil(8+L)/R\rceil$
		33	Short	$97+\lceil(8+L)/R\rceil$
		22		$96+\lceil(8+L)/R\rceil$
	CCK-OFDM	6,9,12,18,24,36, 48,54	Long	$210+\lceil(22+L)/(4*R)\rceil*4$
		6,9,12,18,24,36, 48,54	Short	$114+\lceil(22+L)/(4*R)\rceil*4$

Table 5 - Frame time for IEEE 802.11 PHYs, from [10]

3.4.3 Finding the Available Bandwidth

The previous section shows how to find the busy and access periods for one transaction. In order to find the available bandwidth for a certain period, we sum all busy and access intervals as follows:

$$T_{busy} = \sum_i T_{busy}^{(i)} \qquad (5)$$

$$T_{access} = \sum_i T_{access}^{(i)} \qquad (6)$$

Where $T_{busy}^{(i)}$ and $T_{access}^{(i)}$ are the durations of the ith busy and access intervals. These two times help us figure out more useful and meaningful bandwidth components. A parameter, denoted $\eta_{\max}$, is

needed to convert the time values to bandwidth. $\eta_{\max}$ is also called theoretical maximum throughput (TMT).

$$BW_{busy} = \frac{T_{busy}}{T_{total}} * \eta_{\max} \quad (7)$$

$$BW_{access} = \frac{T_{access}}{T_{total}} * \eta_{\max} \quad (8)$$

$$BW_{free} = \frac{T_{total} - T_{busy} - T_{access}}{T_{total}} * \eta_{\max} \quad (9)$$

Here BW_{busy} represents the portion of the maximum throughput used for the transmission on the load radio. The access bandwidth BW_{access} is used to contend for access the medium, and free bandwidth BW_{free} refers to the unused idle bandwidth. For other stations, idle bandwidth is comprised of two components, respectively BW_{access} and BW_{free}.

3.5 Related Work and Tools

Paper [8] describes a wireless traffic probe for WLAN where the 802.11 MAC the bandwidth components are defined. This give a framework for precise measurement of the WLAN channel utilisation by looking at the MAC frames.

Paper [7] discusses the challenges with wireless sniffers, including location of sniffers and merge of data from multiple sniffers into one stream for processing.

In paper [5], the authors give background and overview of existing monitoring tools, and give another description of channel busy time estimation. One interesting aspect is the use of access points as passive sniffers, instead of using PCs with wireless cards.

The idle and busy bandwidth can be measured either by an active method as described in paper [4] or by a passive sniffing as described in [8]. We will consider both methods. A recent comparison of the passive and active method can be found in [9].

In [3] the application of [8] is described. This is a commercially available tool that very much gives the functionality we are looking for. It uses a wireless card with the MadWifi [20] drivers in monitor mode to collect WLAN data for a single WLAN channel. What it does not support is sniffing in an active access point.

MadWifi is the default driver for WNIC cards with the Atheros chip set. “MadWifi is one of the most advanced WLAN drivers available for Linux today. It is stable and has an established user base. The driver itself is open source but depends on the proprietary Hardware Abstraction Layer (HAL) that is available in binary form only.” [20]

In order to have control of the current spectrum conditions when doing tests, so that results are reproducible, [26] can be used. “Wi-Spy is the world’s smallest 2.4 GHz spectrum analyzer”. This

gives continuous pictures of the radio spectrum conditions, and it is possible to log the results to file for later playback.

For measurements of end-to-end performance we can use Iperf [17]. “Iperf is a tool to measure maximum TCP bandwidth, allowing the tuning of various parameters and UDP characteristics. Iperf reports bandwidth, delay jitter, datagram loss.”

We have selected the latest Ubuntu as our Linux distribution [23].

For monitoring on a Linux PC a wireless card from Atheros [16] together with the madwifi drivers [20] could be used. “MadWifi is one of the most advanced WLAN drivers available for Linux today. It is stable and has an established user base. The driver itself is open source but depends on the proprietary Hardware Abstraction Layer (HAL) that is available in binary form only.”

For monitoring of the current WLAN activity [19] will be used.
“Kismet is an 802.11 layer2 wireless network detector, sniffer, and intrusion detection system. Kismet will work with any wireless card which supports raw monitoring (rfmon) mode, and can sniff 802.11b, 802.11a, and 802.11g traffic.

Kismet identifies networks by passively collecting packets and detecting standard named networks, detecting (and given time, decloaking) hidden networks, and inferring the presence of non-beaconing networks via data traffic.”

Kismet is also able to log and store packets on the pcap format [21]. In preliminary experiments we have managed to log various WLAN MAC layer packet types, but the acknowledge packets comes up in Wireshark as “Malformed” packets. This must be investigated further.

For logging of packets we can use [25]. “Wireshark is the world’s foremost network protocol analyzer, and is the de facto (and often de jure) standard across many industries and educational institutions.” This will give logs in the pcap format [21]. In preliminary experiments we have managed to log WLAN packets, including beacon, associate, authenticate and data. The logs include a monitoring header showing information about data rates etc. The acknowledge packets are showed as “malformed packets”. This must be investigated further.

For processing of packets for bandwidth analysis we can build a C application using the tcpdump/libpcap libraries [21] for packet logging and decoding. There is also a corresponding Java library [18], but it is not yet clear whether this library has WLAN support. The application can either work directly on the logged interface or take log files of pcap format as input.

A Windows alternative for logging is to get hold of an AirPcap device [15]. “AirPcap Classic is the first open, affordable, and easy-to-deploy WLAN (802.11b/g) packet capture solution for the Windows platform.” This is supposed to integrate seamlessly with Wireshark [25]. There is also an 802.11n version available.

For remote monitoring of SNMP MIBs in a working access point MRTG [22] can be used. We have downloaded this and it works “straight of the box”, giving pretty graphs of the traffic load over time. It is highly customisable in that it can graph the values of any SNMP MIBs. One downside is that SNMP monitoring in an access point may take up significant resources that may affect the operation.

4 Collection, Calculation, Visualization and Precision

We divided our project into four sub-problems in Section 1.2 and their solutions are presented in this chapter. Every sub-problem is analysed and solved, and practical problems are handled.

4.1 Data Collection

Initially we spent quite some time on looking for appropriate tools. Details can be found in Section 3.3.The tools found belong to one or more of the following categories:

- Tools providing logging of WLAN MAC layer frames by passive sniffing. This can be done by a PC or an access point in monitoring mode (not operating as access point)
- Tools providing logging and presentation of SNMP data

In addition it is possible to set up a central logging server collecting data from a number of logging clients. The clients can be of different types, Linux PCs, Windows PCs and monitoring access points.

Tools for active sniffing have not been considered further as we quite early made the decision to focus on passive sniffing only.

4.1.1 Getting the Data from the Radio Medium

The tools rely on sniffer hardware to get the data from the radio medium. We have tried different wireless network cards (WNIC). What makes them different is their WLAN chip set, which can come from vendors like Atheros and Broadcom.

The WNIC also needs a driver. This is complicating things further. First we have to find and install the correct driver for the selected WNIC, and then have to put the WNIC into monitor mode. The current Linux distributions now come with support for major WNICs, but monitor mode is not plug and play, and there is little documentation, so it is always a struggle to get a new WNIC up and running.

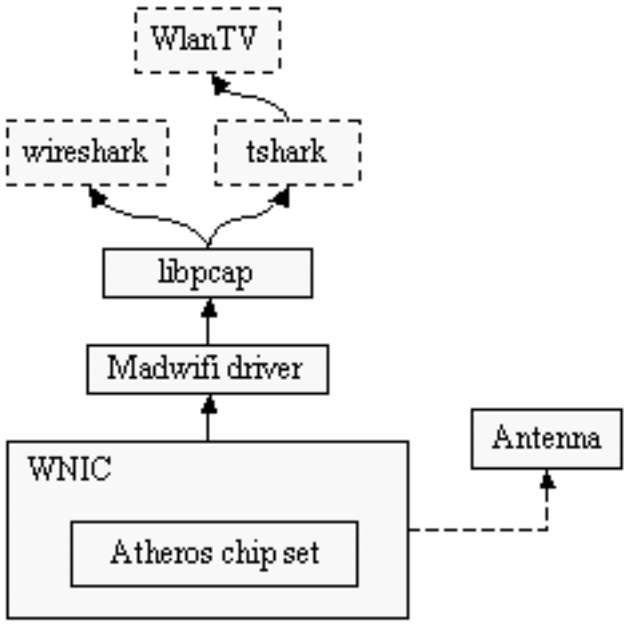


Figure 11 - Sniffing driver hierarchy

With the WNIC and driver in place the next is to get the collected data into the computer. Initially we used kismet [19], however the capture files only contained low rate packets like beacons and probe messages, so we tried Wireshark [25] instead. This gave better results, but still the ACK packets were flagged as invalid. This is a typical situation with the Linux WLAN sniffing. It works partly. We tried different cards and different drivers, but always with some sort of problems.

We then realized that the drivers could produce capture files with different WLAN header types. This is the internal header containing the information from the WLAN PLCP header. The previous de facto standard was the Prism header that comes as default with the Madwifi [20] drivers.

The current and improved header type is the Radiotap header. When we reconfigured the Madwifi driver to use this header, we finally got the logging under control. It is still a manual procedure to set up the logging every time we turn on the computer, but it works.

The documentation of the Radiotap header (and Prism header) in combination with the chipset is non-existent. This meant we had to do a lot of logging just to be able to interpret the different fields. It also turned out that some of the fields couldn't be trusted.

- With Atheros cards/Radiotap the preamble information could not be trusted, and we had to find a workaround. How we solved this is explained later.
- The channel mode information was not documented and we this has been one of our main issues.
- We also get logs from other sources with the Prism header. The problem is then that the Prism headers in these log files is missing channel mode. We are giving a limited support to Prism header log files.
- From some logs made with Prism headers we have also experienced invalid packet length values.

The Windows alternative using the AirPcap [15] USB sniffer has also been tried. This produces log files with radiotap headers and is much easier to set up than the WNICs on Linux. However, a number of limitations have been found with this device:

- It reports wrong preamble length like the Linux WNICs
- It captures less packets than the Linux WNICs
- We couldn't make it work on channel 13
- It was unstable when logging in a setup with a few 11b stations

As the Linux WNIC sniffing turned out to work very well we did not try configuring an access point as a sniffer.

Our primary capture choice for the experiments ended up being a Linux Ubuntu computer with a 3Com WNIC the latest version of the Madwifi drivers for Atheros.

4.1.2 Parsing the Captured Data

We tried to get hold of the software of [3] which would give pretty much of what we are looking for, but this was simply too costly for our budget. We also tried to get into contact with the authors of [5] who has another tool, but we did not get any response. As we found no free tools giving the possibility

to parse the logs and identify the WLAN bandwidth components, we had to design one ourselves. We call it the "WLAN Traffic Visualizer", or WlanTV for short. The tool is described in Chapter 5.

There are different tools and libraries that can be used to collect the data from the WLAN drivers. What they have in common is that they all rely on the libpcap [21] C library or a variant of this. This gives raw network layer packets needing additional dissectors for every protocol to be supported.

At this time we had chosen Java as our preferred implementation language and platform, so we had to find a way to get the pcap packets into the Java program. The jpcap libraries [18] support pcap capture in a Java program. However, these libraries lack the WLAN dissectors. So by choosing jpcap we would have to make a complete dissector for the WLAN protocols, which would be too much for our time frame.

This is where Wireshark, or its command line counterpart, *Tshark* comes in. Wireshark/Tshark actually has dissectors for most communication protocols, if not all, including WLAN. It is also actively maintained and supported, which means it will also support newer standards coming up.

The idea was then to make our application use the Wireshark dissectors in some way. There are a few ways to do this. One way is to write use the Wireshark *taps* interface and call the dissectors through this, but this requires the odd Lua scripting language, which does not fit well with our choice of Java. A better solution is to run Tshark with appropriate parameters to execute the WLAN dissectors on the capture file, in order to produce a text output with the decoded WLAN packets. Tshark can be executed from within the Java program and the command output is directed back to a Java process.

Tshark can be told to decode pcap in different ways. One option gives a textual log exactly like the information seen in Wireshark. Another option is to produce an XML file with tags for all fields. The third option is to give Tshark a list of selected fields for parsing. We tried all the three ways. The XML format is supposed to be good for data representation, but in this context it gives too much data and the parsing is far too slow. We got somewhat better performance with the pure text file and, surprisingly, the parsing was easier. However, the best performance we obtained by giving the exact WLAN field list to Tshark. This also turned out to be a very flexible solution, i.e. it was very easy to adapt the field list to the needs as we advanced with our tool implementation.

Example of generated Tshark command, with field list:

```
tshark -E separator=! -r log.cap -T fields -e frame.time_relative -e frame.number -e
frame.len -e frame.protocols -e wlan.fc.type_subtype -e wlan.fc.retry -e wlan.duration
-e wlan.sa -e wlan.da -e wlan.ra -e wlan.ta -e wlan.bssid -e wlan.frag -e
wlan.fc.moredata -e wlan.seq -e wlan.fcs_bad -e prism.channel.data -e prism.rate.data -
e prism.frmlen.data -e radiotap.channel -e radiotap.datarate -e radiotap.length -e
radiotap.mactime -e radiotap.flags.preamble -e radiotap.channel.type -e
wlan_mgt.fixed.capabilities.short_slot_time -e wlan_mgt.fixed.capabilities.preamble -e
wlan.fc.tods -e wlan.fc.fromds -e radiotap.dbm_antsignal -e wlan.flags -e wlan.fc.ds -e
prism.signal.data
```

4.1.3 Sniffer Location

Even if we deploy a basic WLAN that has only one access point, the distance between the sniffer and the stations/access points is critical for the quality of the collected data. There are a number of cases that ideally should be considered in the experiments:

- The sniffer is placed close to the AP.

- The sniffer is not placed at the AP.
- The stations are placed with some distance between each other. This means we will experience the hidden terminal problem. This case may be extended by using more than one sniffer.
- Sniffing inside of the AP. So far we have not identified any tools that can do MAC layer logging in an active AP. There is however other data, collected by SNMP, that can be used to collect statistical data, like packet counts, retransmission counts, etc that can be correlated with the data from the passive sniffers.

Through the experiments, we can analyse:

- What is the distinction between different cases?
- Too what extent are we able to precisely find the bandwidth components for the different cases?

We did not have enough time to go into this, we however we experienced sniffer location problems when doing our experiments as we shall see later.

4.1.4 Other Issues

Already in our initial experiments we experienced inaccuracy and jitter in the time stamping of the incoming packets. This causes overlapping packets, as we shall see later. The problem appears with all WNIC and driver combinations we have tried so far, and also with the AirPcap sniffer. It is not clear what is the reason. However, the packets come in the right sequence so it is not preventing us from making a correct representation of the packet train.

4.2 Calculation of the Bandwidth Components

We have then reached to core problem of this thesis. Initially we were fascinated by the work in [8], which gives a good introduction to the area. A mathematical model is made to represent the different bandwidth components. There are random components in the calculations like the deferral periods and backoff times that are handled by finding factors through offline calculations, based on how many stations are contending. However, this work has a major flaw. It does not consider *transactions*. As the contention for the medium only takes place at the start of a transaction, and not inbetween the packets with a transaction, the access time must only be counted per transaction.

Paper [5] introduces the use of transactions. This work describes another way to find the access times, by looking at the spaces between the transactions, using mean values for backoff times and corrective factors. See Section 3.4.2 for details.

We planned to implement the method described in [5], but after our initial experiments and further reading of the 802.11 standard [31], we realized that also this method has serious shortcomings. The standard is not clear with respect to in what situations the transactions are preceded by a deferral period, and the different WLAN equipment vendors have interpreted the standard in different ways. As an example, when one station is alone on the medium, there is no deferral and it is supposed to go directly into backoff after a transaction, see Section 4.4.1.3 for details. This is just one example of what is not considered in [5].

Moreover, the above two papers have only partly handled the retransmission and collision/corrupted frame cases. This information is simply not available on the medium, although it may be found in some cases. The backoff times are increased in case of retransmissions and collisions. However, it is only the station itself that knows about the retransmission or collision, and during the extended backoff times, other stations will be allowed to use the medium. It is therefore our assumption that the medium can be considered free during the extended backoff times.

It would require a huge effort to establish an improved and correct mathematical model for the bandwidth components, and even harder to prove it by experiments, given the imprecise standard and the different WLAN equipment vendors all with different interpretations. This has led us to choose a more practical approach for the calculation of the bandwidth components, as described below.

4.2.1 Finding the Bandwidth Components

The main idea of our approach is to extend the work based on [5] in that we identify all transactions. The following assumptions and simplifications have been added:

- Aborted and corrupt transactions and retransmission transactions are handled like other transactions.
- All transactions are preceded by one deferral period DIFS.
- All transactions are preceded by one backoff time.
- The backoff time is fixed to CWmin/2
- The DIFS and CWmin values are deducted from the channel mode of the first packet in the transaction

Given the above conditions all transactions can then be processed and the bandwidth components can be found

> *Bandwidth component calculation*
> The *busy time* is the sum of the duration of all the transactions. The *access time* is the sum of all the deferral periods and backoff times for all the transactions, but not more than the free space between the current transaction and the previous transaction. The *free time* is then the remaining time.

When we have found the free time the next step is to find the potential extra load that can be added to the channel. The free time is not directly available for additional transactions as it is fragmented, and a significant part of the fragments have a size that is smaller than a single transaction. Depending on the packet size of the additional transactions we will also get different results, partly because smaller size means that more small free fragments can be used, and partly because larger packets will introduce less overhead.

We have implemented two different ways to find the potential extra throughput, given a frame size, channel mode and rate of the additional transactions:

> *Optimistic available bandwidth calculation*
> We take the total free time and divide it by the duration (including access time) of one added transaction. This will give a potential added transaction count and easily translates into a throughput figure.

Pessimistic available bandwidth calculation
We find all spaces between the transactions and fill in transactions with access time where the space is big enough. The added transactions are counted and the available throughput can be found.

In both methods the added transaction consists of a unicast frame including the duration of the corresponding acknowledge message. This makes a complete transaction with respect to the duration calculations.

4.2.2 Finding the Transactions

The list of our supported transaction types is given below, in EBNF format. The list is a subset of the transactions types given in the 802.11 standard [31], Section 9.12, but all transaction types under the DCF access method are included.

broadcast	Frame RA is the broadcast address
frag	Frame has its More Fragments field set to 1
group	Frame RA has i/g bit set to 1
individual	Frame RA has i/g bit set to 0
last	Frame has its More Fragments field set to 0

(* This rule defines all the allowable frame exchange sequences *)

`frame-sequence =`

([**CTS**] (**Management** +*broadcast* | **Data** +*group*)) |

([**CTS** | **RTS CTS** | **PS-Poll**] {frag-frame **Ack**} last-frame **Ack**) |

(PS-Poll Ack))

(* A frag-frame is a non-final part of an individually addressed MSDU or MMPDU *)

frag-frame = (**Data** | **Management**) +*individual* +*frag*;

(* This is the last (or only) part of a an individually addressed MSDU or MMPDU *)

last-frame = (**Data** | **Management**) +*individual* +*last*;

The transaction duration can be calculated in different ways. The theoretical approach would be to separate the individual packets with SIFS and let the sum of the SIFS (zero or more) and the packet durations give the total transaction length. The practical approach would be to take the time difference between the start of the first packet and the end of the last packet as its duration. Given the timestamp jitter that we are experiencing in our experiments, the former is our preferred method. However, the NAV should also be considered here, as this is what the other stations will perceive as the busy time. Every packet in a transaction except the last packet carries the expected duration until the end of the transaction, or at least until the end of the next packet. Our selected method is then as follows:

Transaction duration calculation
The transaction duration is the duration of all packets in the transaction except the last, plus the NAV field of the second last packet, plus the sum of all SIFS in the transaction except the last. In case of a one-packet transaction the transaction duration equals the packet duration.

4.2.3 Finding the Packet Durations

The packet duration is dependent on the channel mode used, the slot time, the data rate and the preamble length. The channel mode and the data rate are directly available from the radiotap header, so that is the easy part. The preamble length is also in the radiotap header, but our experience is that this information cannot be trusted. The slot time is not available in every packet.

The current slot time is broadcast in the beacon messages. It can be either short or long.

The preamble length can be short or long, but the beacon only tells whether short preamble is allowed or not. However, it is our experience that whenever short preamble is allowed it is also used by the access point and all the stations. This can be verified by checking that the NAV/duration values of the sent unicast packets match the duration of the acknowledge packets. More details about this can be found in Section 4.4.1.1.

In order to find the slot time and preamble time for a certain packet, we first need to relate every packet to a certain access point/link. We then update every packet's slot time and preamble time with values in the last beacon on the link. We must always use the current beacon message as the slot times and preamble times may change as stations enter or leave the network.

When the four parameters are available, the TxTime formulas from the 802.11 standard [31] can be used to calculate the packet duration, including the duration of the preamble and the PLCP header. We have taken the compiled list of formulas in Table 5 from [10].

Interpreting the channel mode is not as simple as it may look at the first sight, as there is no direct relation between the channel mode indicators in the radiotap header and the 802.11 standard. In particular we have had problems grasping the meaning of the "mixed mode" which is a setting that is available in most access points. We have observed at least two flavours of the "mixed mode":

11g with protection
This is the protected 11g mode (either CTS or RTS/CTS) described in the 802.11 standard. In this case all packets are marked with channel mode "pure11b" or "pure11g". (The 11g packets are proceeded with a CTS or RTS/CTS in 11bmode).

11g mixed mode
In this mode all packets are marked with "11g" and the data rate indicates whether it is an 11b packet or an 11g packet. Data rates 1, 2, 5.5 and 11 indicate 11b while rates 6,12,18,24,36,48,54 indicate 11g.

4.3 Visualization of the Traffic Load

The basis for the visualisation is the following information:

- A list of transactions and packets, including timestamps, durations and packet lengths
- A list of active links/connections between stations and access points

4.3.1 Train Chart

Very early in this project we made a prototype in order to find some good representations of the WLAN traffic, and then what we call the *train chart* came up.

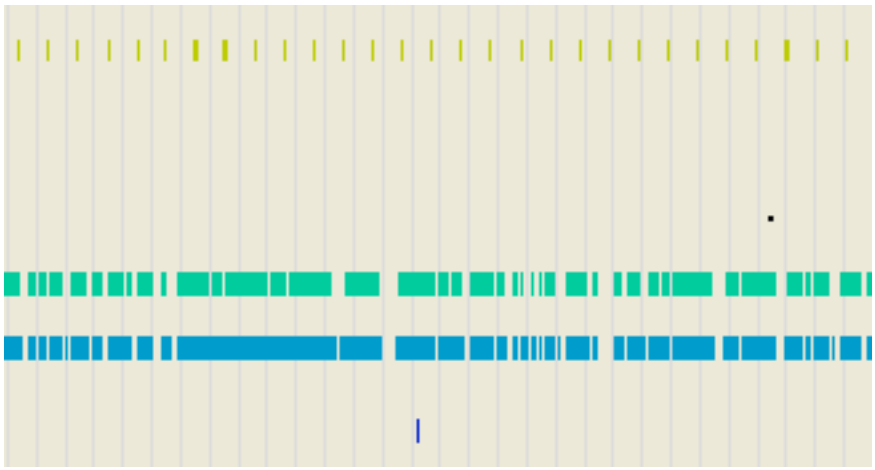


Figure 12 Train Chart, overview zoom

The train chart shows the packets with the time on the X-axis and the packet type on the Y-axis. The packet types are located according to their sub type number in the 802.11 standard with the management packets at the top, the control packets in the middle, and the data packets at the bottom. The packet types are also colour coded for convenience. In Figure 12 the beacons are light green, acknowledge packets are sea green and data packets are blue. The vertical lines show the horizontal time scale, and in this case the time between the lines is 100ms.

The chart can be zoomed in and packet details and transactions are revealed, see Figure 13.

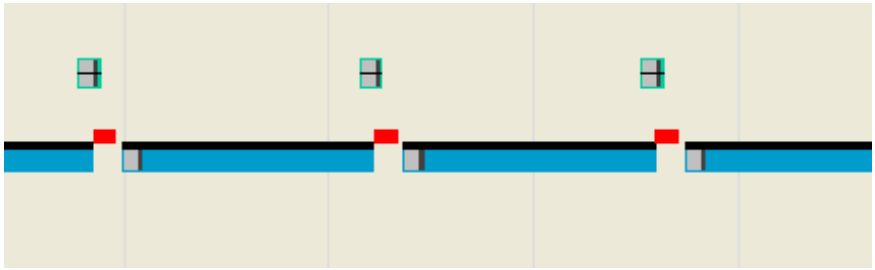


Figure 13 - Train Chart, packet zoom

The colour coding is still the same, but PLCP preamble and headers are indicated with light grey and dark grey colours respectively, and the duration field (NAV) of the data packet is coloured red. Uplink packets are indicated with a black horizontal line on the top of the packet while downlink packets have the line at the bottom. The acknowledge packets have no defined direction, so the line is placed in the middle.

There is additional information in this chart that will be introduced in Section 5.2.3.

The train chart has proved to be an extremely useful visualization of the WLAN traffic, both for understanding and verifying packet sequences, and for the verification of our method.

4.3.2 Other Charts

Pie charts and a load chart are also available. Please refer to Sections 5.2.6 and 5.2.7 for details.

4.4 Efficiency and Precision

4.4.1 Basic Sniffing

The train chart plays a major role in the verification of our method.

By visualization of collected data it is possible to tell whether the sniffing was successful. A quick glance at the train chart reveals what packet types were transmitted, and by applying a filter (see 4.4.3) it is easily seen what access points and stations that have been active for the period. The colour coding and position information in the train chart makes it easy to verify that the test was run as planned

Moreover, by zooming in so that only a few transactions are seen, the timestamps and durations can be checked. At this point, we have been struggling with two problems. See Figure 14 and Figure 15.

Packets overlap

The MAC part of the can packets overlap, but more frequent is the packet overlap after the addition of the PLCP preamble and headers.

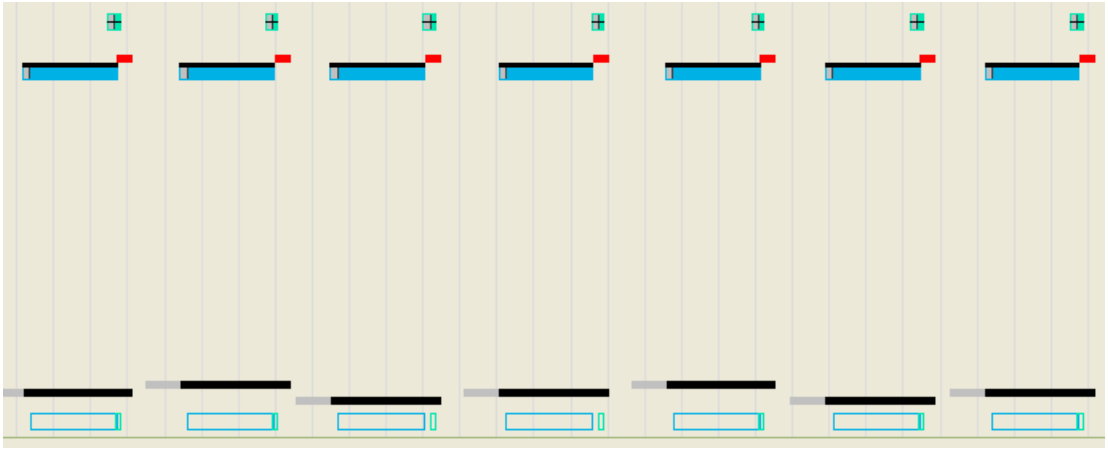


Figure 14 – Example of packets overlap in 11g

Transaction Overlap

After adding the deferral period and backoff time to the transaction we normally see overlaps. If the overlap over time averaged to zero this would be ok, as our estimated backoff time is estimated to be the mean value of the initial backoff time, that is CWmin/2. However, we often see far greater transaction overlap than this.

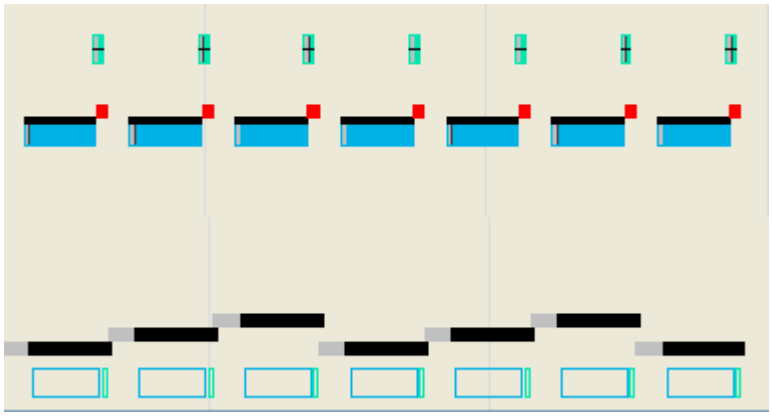


Figure 15 – Example of transactions overlap in 11g

The overlap problem has a number of causes:

4.4.1.1 Inaccurate Time Stamps

The packet timestamp from the driver is supposed to be taken when the last byte of the MAC layer data is received. Our experiments confirm this. However, the time between the timestamp of a unicast packet and that of the following acknowledge packet is often less than the duration of the acknowledge packet itself, and this clearly shows that the timestamp is not accurate. We have looked into this, but we have not found where the problem is introduced. If it is so that the timestamp originates from the WNIC card, it may be that the time stamping function is simply not prioritized as it is may not be critical for the basic WLAN operation.

To monitor the timestamp inaccuracy we have introduced a function that aggregates all the delays between data unicast packets and the following acknowledge packets for every link, and we calculate the mean value and the standard deviation of the delay. This way we can keep the inaccuracy under control, and see how it changes with different sniffing solutions.

Similarly, we find the average and the standard deviation of the NAV of all unicast packets. The difference between mean values of the measured unicast to ack delay and the NAV is monitored, and if it is above a certain limit, a warning will be given.

4.4.1.2 Wrong Channel Mode

In order the find the duration of packets, transactions and access times, we need the correct information of the used channel mode. This information is only partly available from the sniffed data. The channel mode is made up of the channel type from the PLCP header, the preamble length and the slot length. When collecting the slot and preamble length from the beacon as described in Section 4.1, we get the channel mode right in most cases.

In case a station or an access point uses a wrong channel mode, or if we of some reason have detected the wrong channel mode, this will normally be detected by the monitoring functions as described in Section 4.4.1.1.

4.4.1.3 Stations or Access Points with Different Interpretations of the Standard

The space between the transactions varies depending on mode, data rate, WNIC brand, access point brand. We have not made a survey of this, but we have some observations.

The 802.11 standard is not very clear when it comes to when to insert the DIFS and backoff time. This is a clip from [31] clause 9.2.5.1:

> "In general, a STA may transmit a pending MPDU when it is operating under the DCF access method, either in he absence of a PC, or in the CP of the PCF access method, when the STA determines that the medium is idle for greater than or equal to a DIFS period, or an EIFS period if the immediately preceding medium-busy event was caused by detection of a frame that was not received at this STA with a correct MAC FCS value. If, under these conditions, the medium is determined by the CS mechanism to be busy when a STA desires to initiate the initial frame of one of the frame exchanges described in 9.12, exclusive of the CF period, the random backoff procedure described in 9.2.5.2 shall be followed. There are conditions, specified in 9.2.5.2 and 9.2.5.5, where the random backoff procedure shall be followed even for the first attempt to initiate a frame exchange sequence."

According to this it seems that the backoff procedure shall not be used in the case the medium is free for more than one DIFS period (in the normal case) after a transaction.

We also understand that the backoff procedure does not include the DIFS.

Another clip, from Section 9.2.5.2 of [31]:

> "A backoff procedure shall be performed immediately after the end of every transmission with the More Fragments bit set to 0 of an MPDU of type Data, Management, or Control with subtype PS-Poll, even if no additional transmissions are currently queued. In the case of successful acknowledged transmissions, this backoff procedure shall begin at the end of the received ACK frame. In the case of unsuccessful transmissions requiring acknowledgment, this backoff procedure shall begin at the end of the ACKTimeout interval (as defined in 9.2.8). An unsuccessful transmission is one where an ACK frame is not received from the STA addressed by the RA field of the transmitted frame and the value of the RA field is an individual address. If the transmission is successful, the CW value reverts to aCWmin before the random backoff interval is chosen, and the SSRC and/or SLRC are updated as described in 9.2.4. This assures that transmitted frames from a STA are always separated by at least one backoff interval."

Here it is stated that the backoff procedure shall be performed immediately after a transaction. *This means that there is no DIFS between transactions in the case when only one station is sending.* This seems to be in contradiction with the previous clip stating that there is no backoff procedure when the medium is free for a DIFS period.

Different interpretations at this point may explain why we experience varying idle times between transactions with equipment from different brands. This also means that it is impossible to make an exact prediction of the free bandwidth without considering the brands of equipment used. We solve the prediction issue by allowing the user to adjust the access time (backoff+DIFS) and the packet length of the added transactions depending on the actual situation.

4.4.1.4 Stations of Access Points with faulty Implementation of the Standard

We also have logs showing faulty implementations, like in Figure 16.

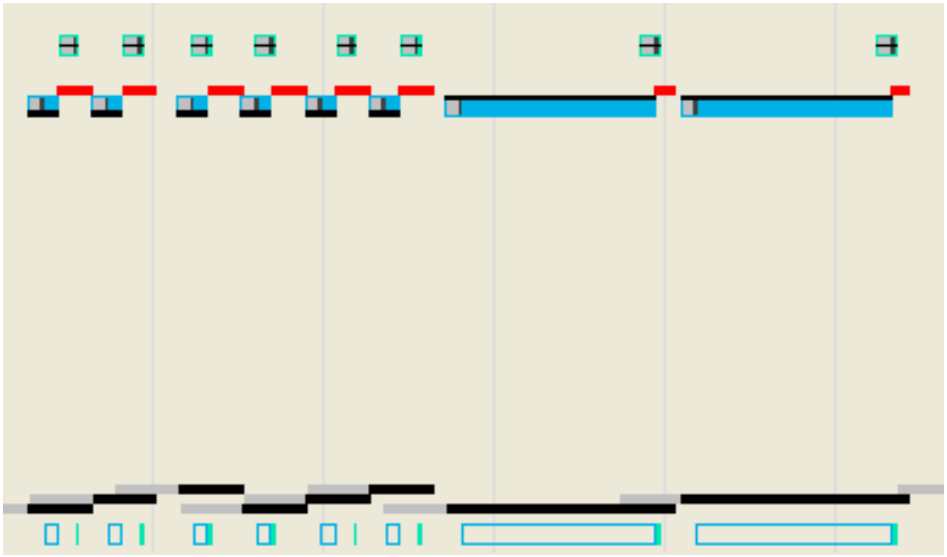


Figure 16 – Example of bad transactions overlap in 11b mode

In this case the beacon is signalling the use of long slot time and long preamble. However, the uplink packets are sent with a NAV indicating short preamble, while the uplink packets from the access point are sent correctly with a NAV value corresponding to the long preamble.

Moreover, there is also a bad transaction overlap. We don't know the reason for this, but it may be that the link is using short slot time even though the beacon signals that this is not allowed.

4.4.1.5 Sniffing Solution

The quality of the collected data is dependent on sniffing solution itself. There are a number of factors:

- The computer hardware used (new PC, old PC, PCI speed, USB speed, hard disk speed)
- The state of the sniffing computer (busy or not, general condition)
- The brand of the sniffer card/USB sniffer and used chipset
- Windows/Linux
- Linux driver and driver version

Some configurations give better results than other. In Chapter 6 our primary sniffing solution and test configuration is described.

4.4.2 Lost Packets

In case packets are lost due to the hidden terminal problem or interference, we have identified two cases that can be handled relatively easy, so that we don't miss the lost packets' contribution to the busy bandwidth:

Aborted Transaction In the case of an aborted transaction, the last received NAV will decide the length of the transaction. This means that the busy time calculation is not affected by the loss of e.g. an acknowledge message, or the DATA after a CTS.

Lost unicast packet
In the case of the loss of a unicast packet and the successful reception of the following acknowledge packet, the lost unicast packet is reconstructed in order to make a complete transaction for the busy time calculation. The reconstructed packets are displayed in the train chart.

4.4.3 Filter Function

To improve the ease of network analysis, and the handling of neighbour network interference in particular, we have introduced a filter function. This means that every network/access point or station can be analysed separately. This filter functions are given below. Any combination of the filters is possible.

Station filter	Only traffic to and from the selected station, including broadcast traffic from its access point is included.
Access point filter	Only transactions belonging to the selected access point is included.
Channel filter	This will exclude all transactions not using the selected channel. This can be useful in case a neighbour network is running on a neighbour channel.
RSSI filter	This will exclude all transactions where the first packet in the transaction has an RSSI below a selected limit.
RSSI pluss Access point filter	This is a special filter with a special purpose: Transactions from a neighbour network running on the same channel as that of the selected AP will be excluded, if the first packet in the transaction has an RSSI value below the selected limit. This means that only neighbour network traffic that is strong enough to interfere with the traffic of monitored network will be counted as busy time.

Table 6 - View Filter

4.4.4 Verification by Experiments

In order to have confidence in the measurements we will have to set up experiments with controlled traffic streams of different types and saturation levels, and compare the measured bandwidth components with the corresponding theoretical models and reference tools.

The selected test scenarios and test configuration are described in Chapter 6.

4.4.4.1 Comparison with Wireshark

Wireshark [25] can generate number of reports from the capture files. As our tool uses Wireshark/Tshark for the parsing of capture files we will have exactly the same input data as Wireshark. The output from our method can be compared with the Wireshark reports, and we should have a match were the data are comparable.

Among the Wireshark reports are:

- Summary
- WLAN conversation list
- WLAN endpoint list
- WLAN traffic
- IO graphs

4.4.4.2 Verification of the free Bandwidth Estimation

To verify the free bandwidth estimation we will do experiments for selected test scenarios where we keep the packet size fixed while increasing the real load. The sum of the real load and the potential added load should remain stable.

4.4.4.3 Comparison with the TMT

We will compare our results with the Theoretical Maximum Throughput [6]. Our reported bandwidth figures should lie well below the TMT values for the selected packet sizes.

4.4.4.4 Check NAV Against Acknowledge Time

The program collects statistics about NAV versus acknowledge time for unicast packets. If the times differs too much there is a warning.

5 The Wlan Traffic Visualizer (WlanTV)

In order to perform the throughput calculations and traffic visualizations we implemented the Wlan Traffic Visualizer program. This is a program reading capture files and providing charts and other useful data. It is a pure Java program that we have written 100% ourselves, only relying on standard Java libraries and Tshark. It is running on both Linux and Windows.

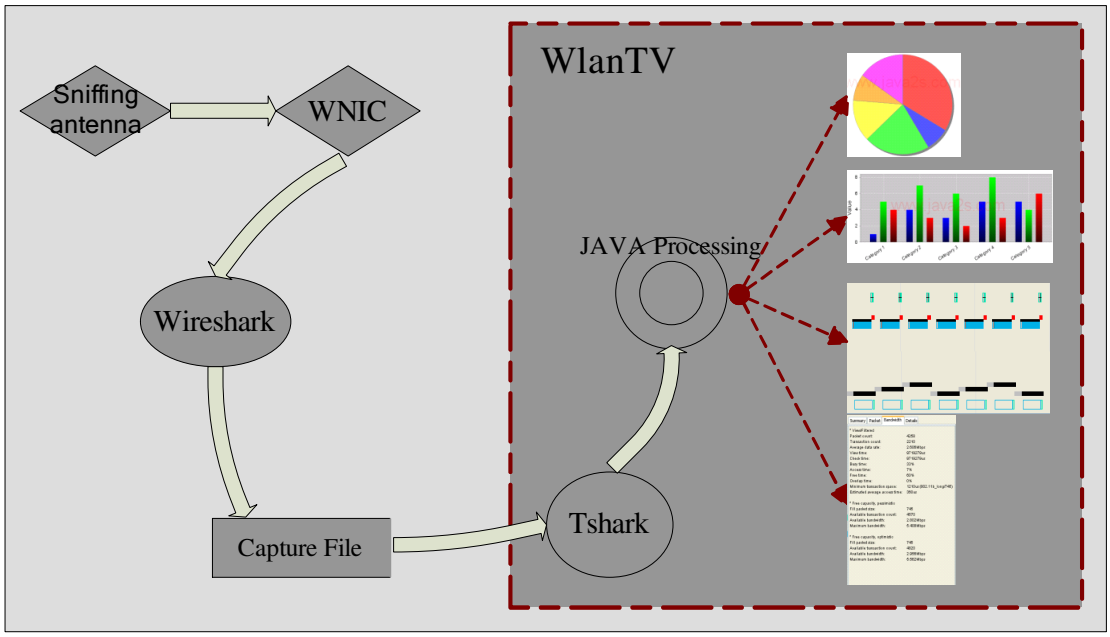


Figure 17 - Implementation Concept

The data collection and analysis has two main steps as shown in Figure 17. The first step is to collect the data, with Wireshark or Tshark and store it to a capture file. The second step is to start the Wlan Traffic Visualizer program with the capture file as input. The capture file is parsed from the Java with the help of the Tshark dissectors, and the results are presented to the user. It is also possible to run the program in live monitoring mode.

5.1 Features

The WlanTV have the following main features:

- It reads WLAN capture files on all formats supported by Wireshark/Tshark
- It can do live capture of WLAN traffic if the computer is equipped with the appropriate sniffer hardware and drivers
- WLAN traffic data is presented in the innovative TrainChart. It gives different information depending on zoom level, and can be used for a number of purposes, including;
    - By feeding the WlanTV with the appropriate logs, the TrainChart can be used for WLAN protocol training and self-study
    - By taking logs from different access points and stations, differences in WLAN protocol implementations are easily discovered and analysed. It can be used for WLAN protocol benchmarking.

- During development of WLAN protocols the protocol behaviour can easily be observed and faults can be removed before they reach the customers.

- Most information panels and filter functions are updated according to the current time range of the TrainChart, which means that the user can select the range with the valid data for analysis.
- Various text panels give details about packets, transactions, used and available throughput, conversations and link information.
- The available throughput estimation can be fully automatic, or manually adapted to the actual situation by tuning access time and length of added transactions.
- The load chart shows the load and available throughput for the capture.
- The pie charts show various counts and time components for the capture.
- MAC addresses can be associated with aliases to simplify reading of packet sequences.
- A number of intelligent filter functions allow the user to limit the amount of data for more efficient analysis.
- The program runs equally well on both Windows and Linux and is based purely on freely available software and libraries.

More details are presented in the following chapters.

5.2 The User Interface

5.2.1 Opening Window

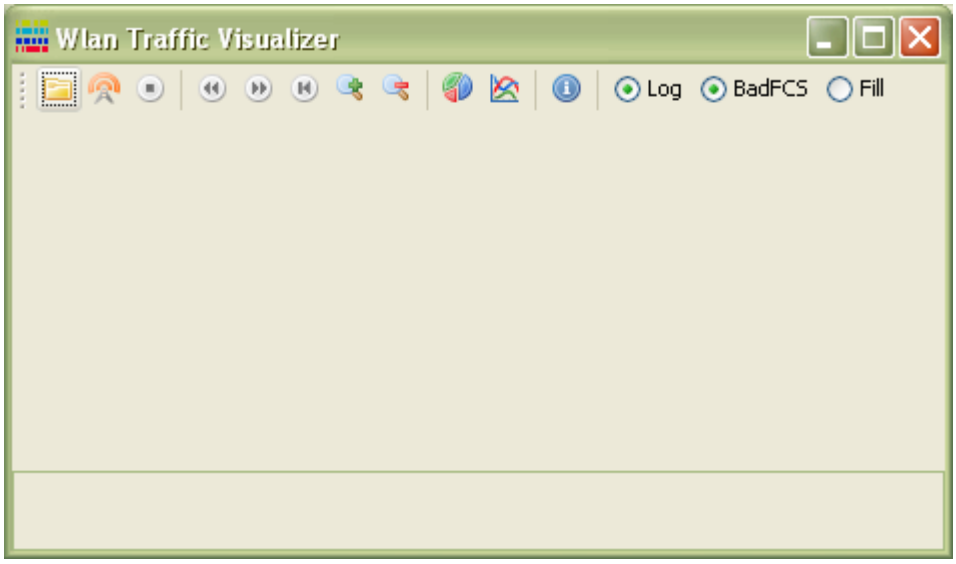


Figure 18 - Opening Window

Figure 18 shows the opening window that is used for opening of a stored capture file for analysis, or starting and stopping a live capture. Then there is a group of navigation buttons and buttons for showing pie charts and a load chart. The radio buttons are used for enabling of warnings, showing bad checksum packets and fill packets. All buttons are explained with tool tips.

The info button gives details about mouse operations – see Figure 19.

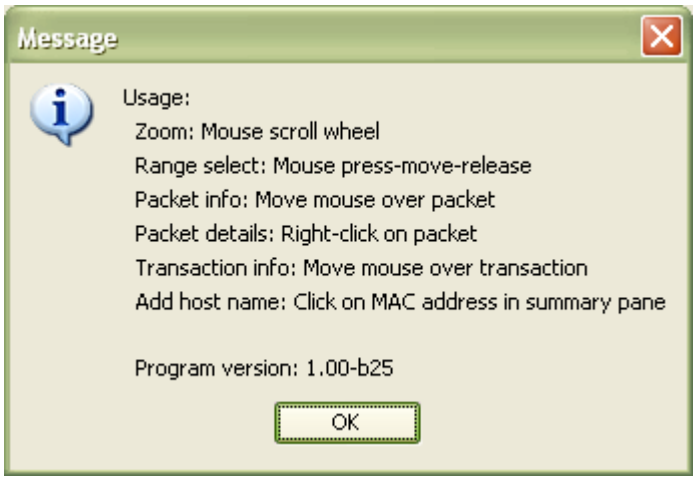


Figure 19 - Usage Information

5.2.2 The Main Window

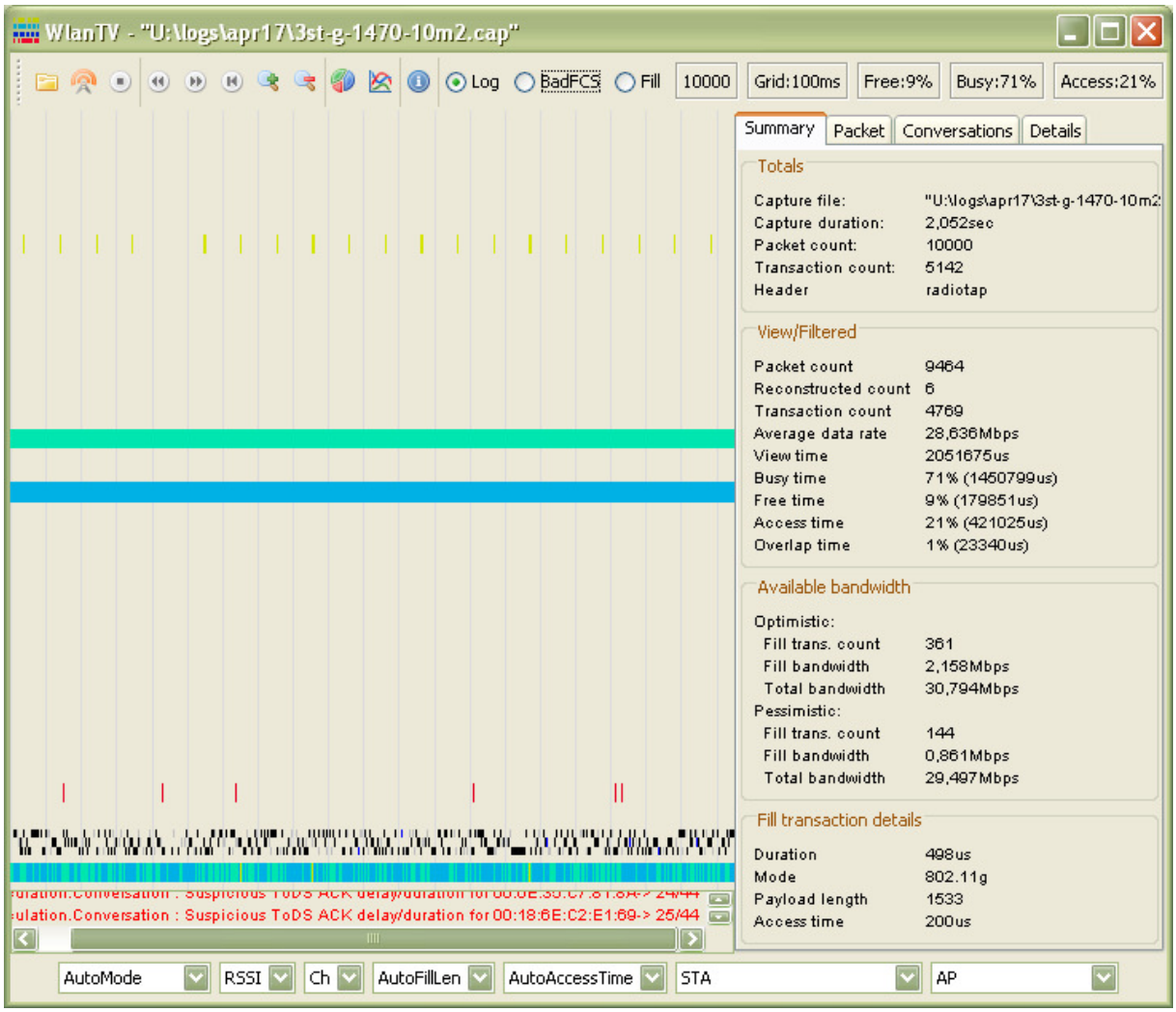


Figure 20 - User Interface Overview

Figure 20 shows the *train chart* to the left and the various panels and other charts to the right. The train chart was introduced in Section 4.3.1, and more details follow below.

The *summary panel* has for sections; the first shows essential and static information from the current log file; the second shows more details, and it is updated according to the current filter and view range; the 3rd shows potential extra throughput according to the optimistic and pessimistic estimation; and the last gives details about the transactions that is added to fill up the channel.

Below the train chart is the *warning panel*, which is used to give warnings and important messages to the user.

The bottom line has a number of drop down menus.

- *Override channel type*. The default is to find the channel type automatically from available frame and link information.
- *RSSI filter*
- *Channel filter*
- *Payload length of data packets added to fill up channel and reconstruct lost data packets.* The default is to use the average length of the packets in the session.
- *Override access time used to fill up channel for free bandwidth calculations.* The default is to use the access time of the first packet in the session.
- *STA filter*
- *AP filter*

5.2.3 Train Chart Examples

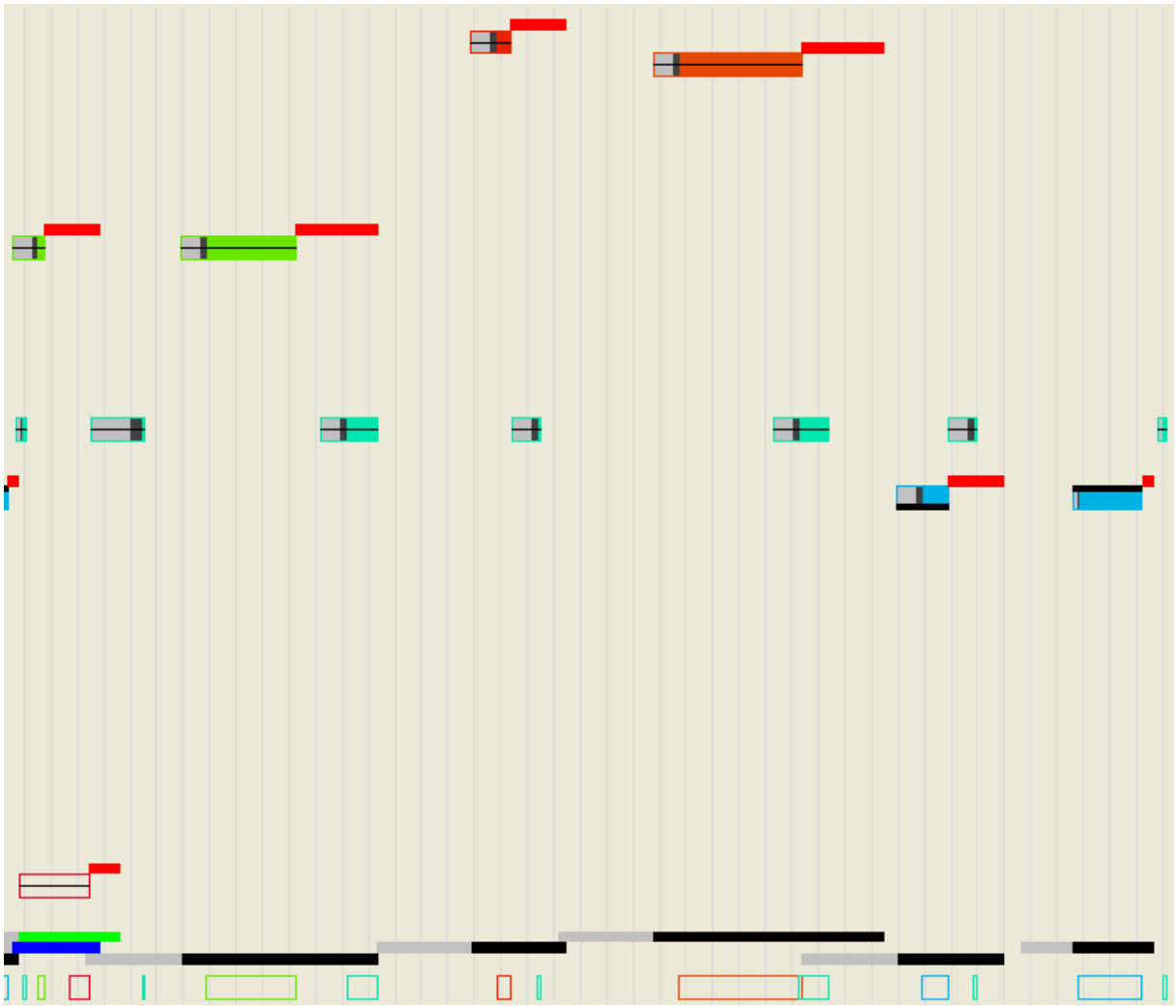


Figure 21 – Train Chart Example – Association

Figure 21 shows an association sequence, with the uplink and downlink authentication packets in green, then the uplink and downlink association packets in red, and finally one downlink and one uplink data packet. Every unicast packet is followed by an acknowledge packet in sea green.

At the bottom the raw sniffed packet information is given in sequence. Above this the transactions are given with the DIFS plus backoff time in light grey colour and the complete packet in black.

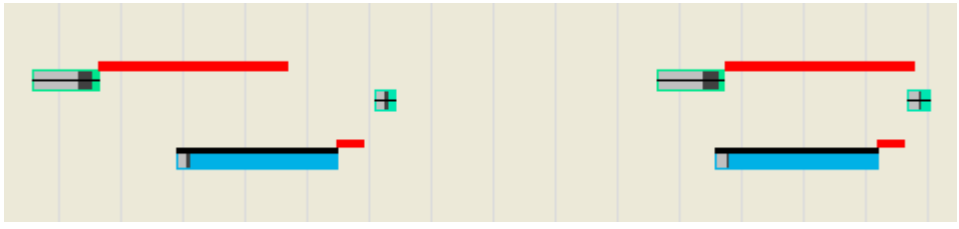


Figure 22 – Train Chart Example – Protection

Figure 22 shows two transactions with CTS-DATA-ACK.

5.2.4 Conversations

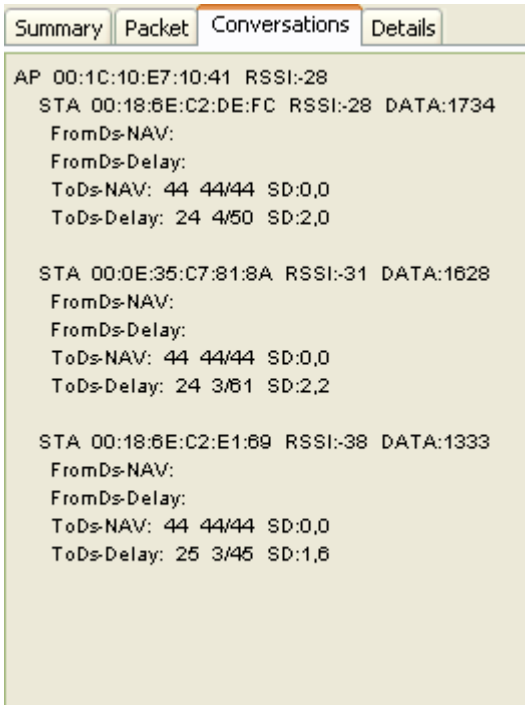


Figure 23 - Conversation Panel - One access point with 3 stations

The *conversation panel* in Figure 23 lists all access points with their associated stations. The link quality of all stations is given as well as the data unicast count for every link.

In addition the average NAV values, with minimum, maximum and standard deviation for all data unicast packets is given. The values are collected independently for the uplink and downlink directions. In the ideal case the NAV value should be the same as the time from the end of the unicast packet to the end of the following ACK packet. The average of the delay is collected in the same way and can then be compared to the NAV value. See Section 4.4.1.1 for details.

By left-clicking a MAC address in the conversation panel it is possible to enter an alias for the MAC address. In the packet panel and conversations panel this alias will be displayed next to the MAC address after the next update.

5.2.5 Packet Information

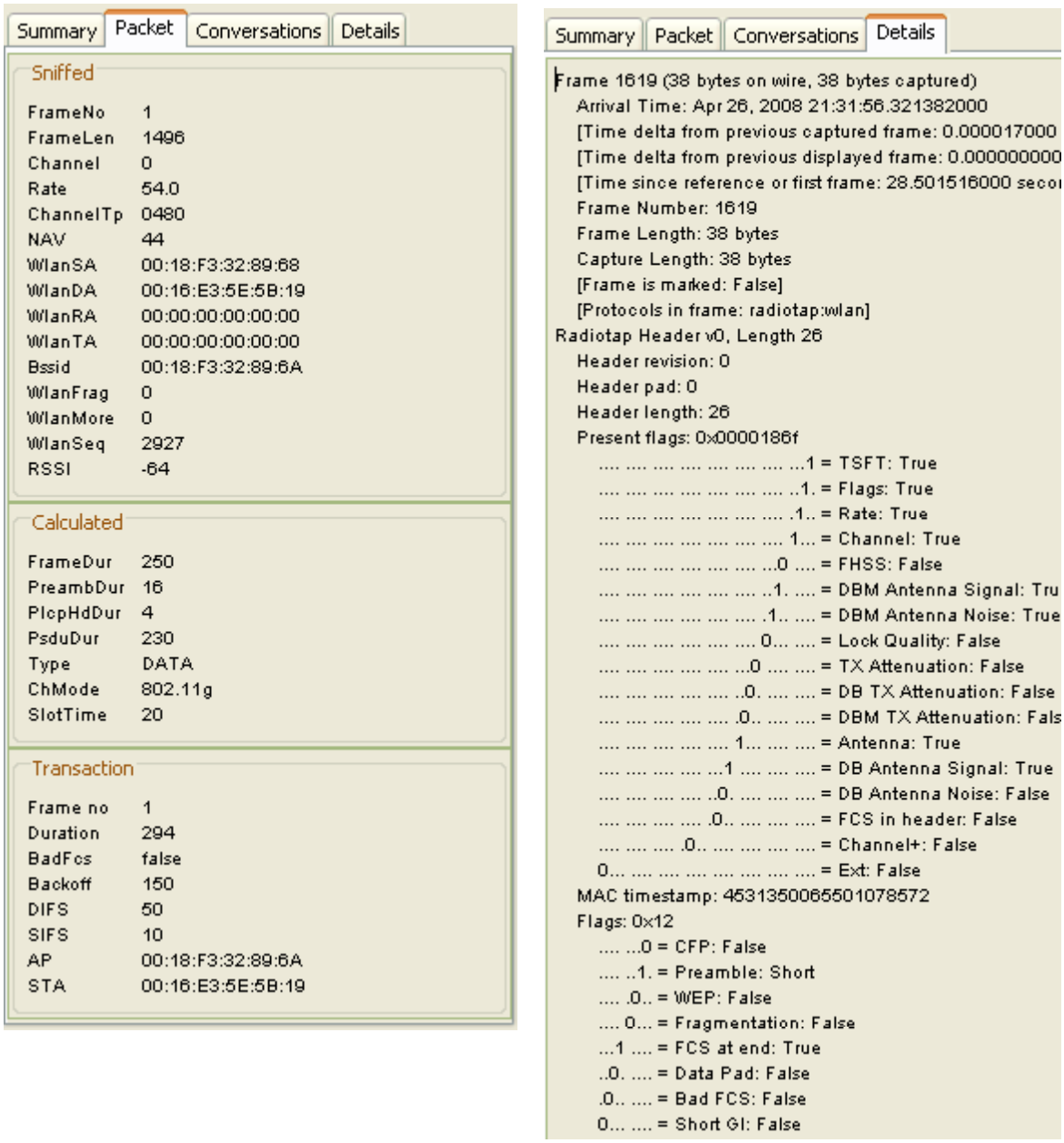


Figure 24 - Packet and transaction details

The *packet panel* in Figure 24 gives information about the current packet and transaction. The information is automatically updated with the data from the packet or transactions that is under the mouse pointer in the train panel.

The *details panel* in Figure 24 gives the complete information for a selected packet as generated from Wireshark. The packet is selected by right-click in the train panel.

5.2.6 Pie Charts

Figure 25 - Pie Charts

The *pie charts* in Figure 25 should be self-explanatory. There are 6 different pie chart types, showing time distributions of frame parts, bandwidth components as well as packet counts.

Please note that the pie chart is updated according to the current view range and filter function.

5.2.7 Load Chart

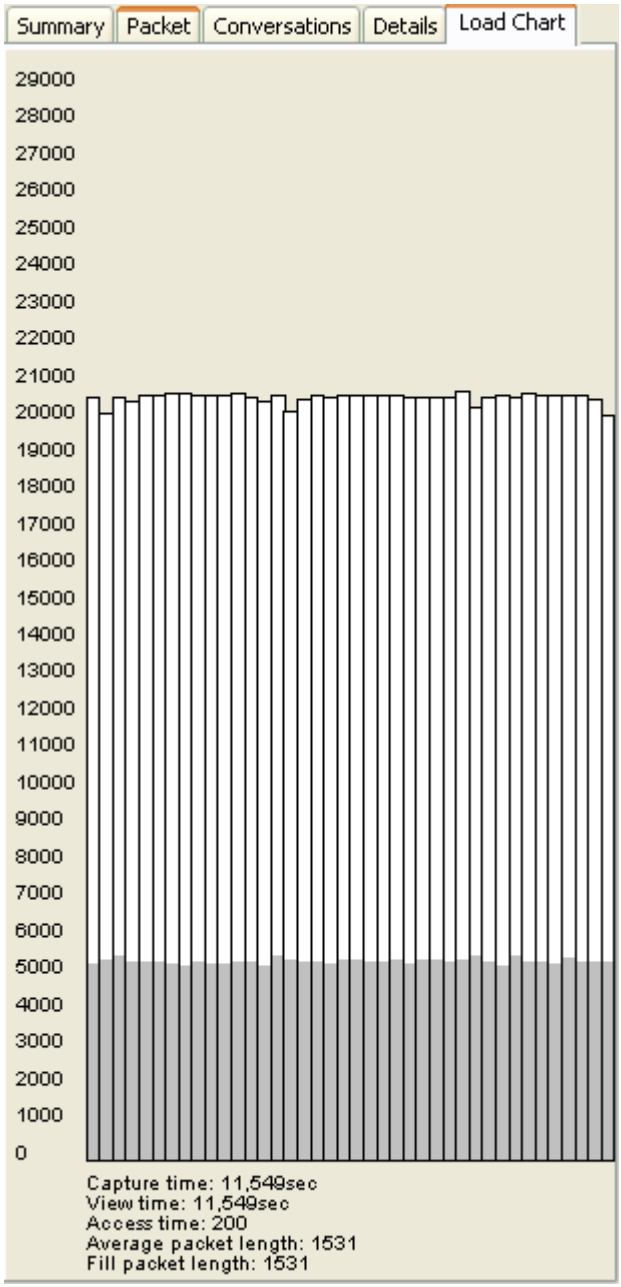


Figure 26 - Load Chart

The *load chart* in Figure 26 shows the used and available throughput over time.

It shows the throughput on the Y-axis in kbit/s (k=1000, not 1024), and the time span for the X-axis is given under the chart. The measured throughput (including reconstructed packets and excluding the access time) is indicated with bars in grey colour and the potential extra throughput is added in white on the top.

In order to find the available throughput the channel is filled up with packets of a configurable length. The “pessimistic” method described in Section 4.2 is used.

The load chart will show the throughput for the current view, which means that it will be affected by the filter function that is described in Section 4.4.3. This way it is possible to get the load for a selected access point or a selected link, for packets with a certain RSSI level, for packets on a selected channel, or for a combination of those.

The following information can be read out of the load chart:

- Total used and available throughput over time
- Throughput per station
- Throughput per access point
- The TMT (see Section 3.1) can be manually calculated based on the current data rate and mode, and compared with the measured and potential throughput in the load chart.
- Please note that visible charts are updated when the view and filter is changed.

5.3 Design

The Java program is divided into three main packages *Collection, Presentation* and *Calculation.* The UML class diagram is given in Figure 27. The package names and contents are aligned with the three first particular problems of this thesis.

- The Collection package takes care of program start-up, data collection and support functions.
- The Presentation package contains all code related user interaction, windows, panels and graphical presentations, and it is the coordinator.
- The Calculation package contains all calculation and statistics functions, and provides data to the presentation functions. Most panels in the Presentation class have a corresponding class in the calculation package.

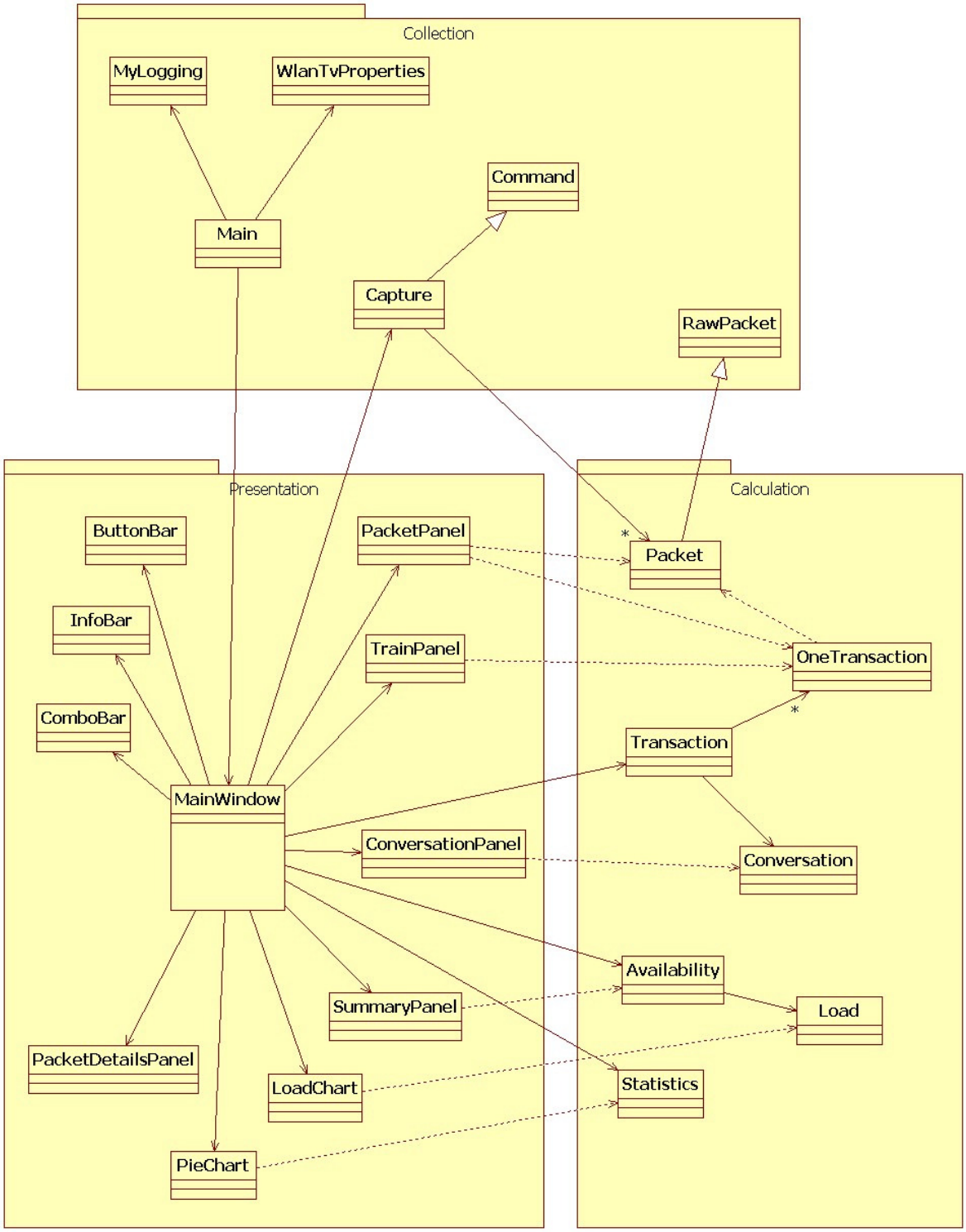


Figure 27 - UML Class diagram for WlanTV

5.3.1 Collection Package

Class	*Function*
Main	Program start-up
Capture	Runs capture from file or live capture Collects packets
Command	Generic class to run shell commands and collect shell response
RawPacket	Holding raw information for single packet Defining what fields to collect by Tshark command Parsing of data collected by Tshark
WlanTvProperties	Handling of program properties for WlanTv program At the first start-up a file *wlantv.ini* is created at the users working directory. The file is initiated with default values. *Properties defined in wlantv.ini* log.dir = updated automatically reconstruct.size = size of reconstructed packets bad.fcs = include | exclude Wireshark.len.compensation = on | off
MyLogging	Handling of warning messages

Table 7 - Functions of the Collection package

5.3.2 Calculation Package

The critical part of the code is located in the Calculation package. Here is the list of the main functions and their location in the code.

Class	*Function*	*Described in Section*
Packet	Calculation of packet duration Holding for calculated information of a single packet	4.2.3
Transaction	Identification of transactions by a finite state machine Calculation of transaction duration Finding transaction addresses Reconstruction of lost packets	3.4.2 4.2.2 4.4.2
OneTransaction	Holding calculated information of a single transaction	-
Conversation	Building of access point and station lists Collection of NAV and delay average for links	4.2.3 4.4.1.1
Availability	Calculation of bandwidth components Fill of extra packets	4.2.1
Load	Aggregates load information for the current view	-
Statistics	Aggregates statistics for the current view	-

Table 8 - Functions of the Calculation package

5.3.3 Presentation Package

Class	*Function*
MainWindow	Coordinating Initiating live capture or file selection Stopping capture
TrainPanel	TrainChart display Live update during collection Setting of view range and zoom Packet selection
ButtonBar	Handling of menu buttons and radio buttons
InfoBar	Showing status information
ComboBar	Handling of combo bars (filters, etc.)
PacketPanel	Showing packet and transaction information
SummaryPanel	Showing summary information from Availability object Adding alias for MAC addresses
ConversationPanel	Showing conversation details form Conversation object
PacketDetailsPanel	Showing packet details from capture file
PieChart	Showing pie chart from Statistics object
LoadChart	Showing load chart form Load object

Table 9 - Functions of the Presentation package

5.4 Supported Standards

The following channel modes, according to Table 5, are supported. (The remaining modes are hardly used according to [11], and common access points do not support them.)

- 802.11 FHSS (check)
- 802.11a OFDM
- 802.11b DSSS and HR-DSSS
- 802.11g ERP-OFDM (This is called the “pure G” mode in Wireshark)
- 802.11g CCK-OFDM (This is called “G” mode in Wireshark)

Due to limited time/equipment the following functions have not been tested:

- 802.11
- 802.11a OFDM

IBSS The following functionality is partly supported:

- WMM: Basic logging should work, but AIFS has not been implemented.
- IBSS

Other limitations:

- Fragments are handled as separate transactions.

6 Test Scenarios and Results

In this chapter, we present test scenarios for the verification of our solutions and showing the performance of our method. We firstly present the test configuration and test scenarios, and then follow the test results, and the last and important analysis of the test results.

6.1 Test Configuration

The basic test configuration is given in Figure 28, with one, two or three stations, and one access point.

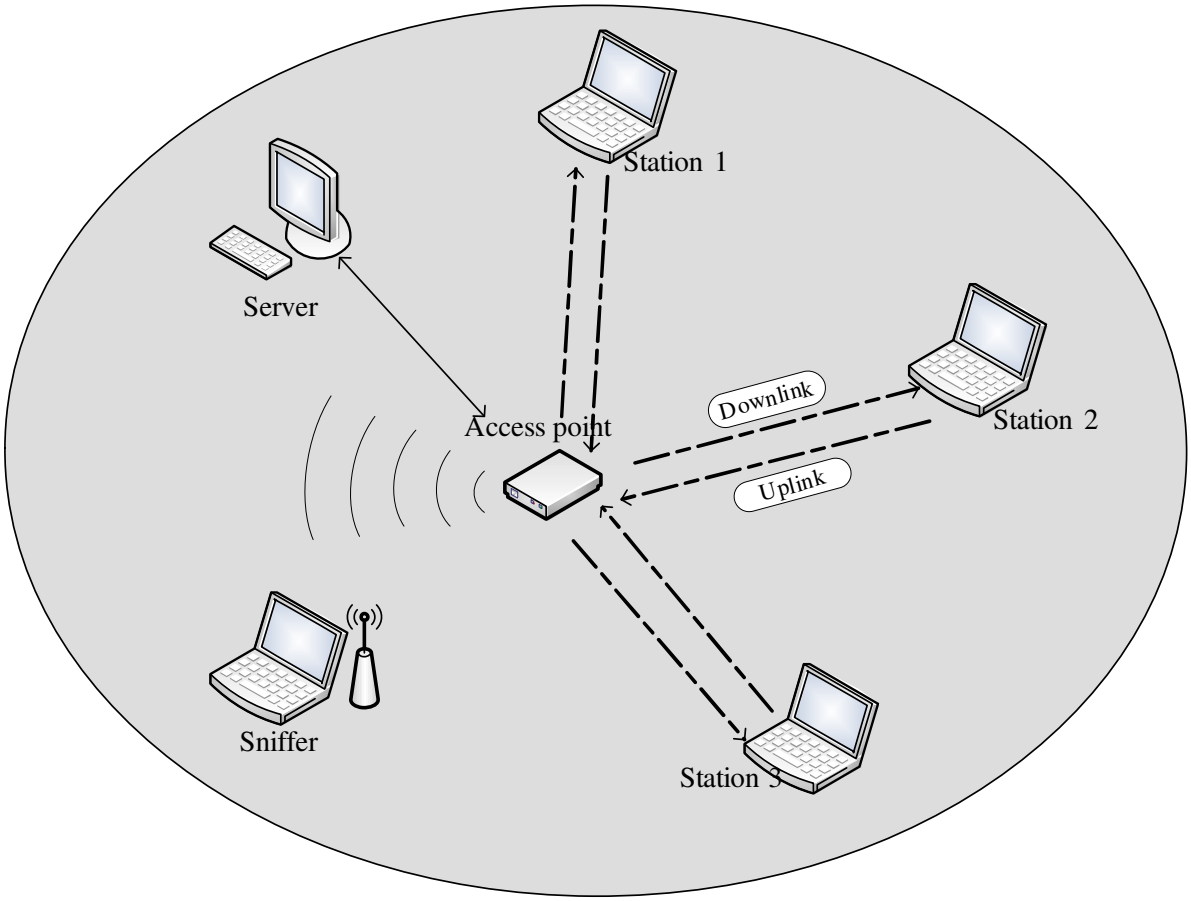


Figure 28 - Test Configuration

- STA1, STA2 and STA3 are standard PCs with a built-in or external wireless network cards. They are used as clients, generating WLAN traffic, typically running the Iperf tool.
- The sniffer is a Linux Ubuntu PC with a wireless card with Atheros chip set and Madwifi driver.
- The AP is the access point. We will only use a single access point.
- The PC connected to the AP can be used as the peer for uplink and downlink traffic, as well as AP monitoring and setup.
- No Internet connection is needed.

- The detailed list of equipment used will be included in the test report.

6.2 Test Scenarios

Rationale for the test case selection can be found in Section 4.4.4.

The following testing conditions apply to all test cases unless otherwise noted:

- The tests are run in interference free environment
- Sniffing packet count 10000
- Sniffing close to access point
- Stations < 5m from access point, in the same room
- Uplink traffic

The test scenarios are given in Table 10.

There are six test scenarios:

(1) Basic TCP test

(2) 1 station, UDP uplink, full load, decreasing packet size

(3) 1 station, UDP uplink, decreasing load, fixed packet size

(4) 3 stations, UDP uplink, full load, fixed packet size

(5) 1/2/3 stations, UDP uplink and downlink, full load, fixed packet size

(6) Access point to mixed mode, one 11b STA and one 11g STA, with protection

The reason of choosing UDP instead of TCP is that TCP will itself adjust the rate at which data is sent out, so TCP is harder to control than UDP. It would make our prediction for channel load inaccurate.

Test scenario	*Mode*	*Rate*	*STA count*	*Traffic Type*	*Length*	*Load*	*Exp. busy pluss access time*	*Exp. free time*	*Exp. pot BW*
(1)	11g	54M	1	TCP	1470	100%	100%	0	0
(2)	11g	54M	1	UDP	-	100%	100%	0	0
(2)	11b	11M	1	UDP	-	100%	100%	0	0
(3)	11g	54M	1	UDP	1470	-	-	-	-
(3)	11b	11M	1	UDP	1470	-	-	-	-
(4)	11g	54M	3	UDP	1470	100%	100%	0	100%
(4)	11b	11M	3	UDP	1470	100%	100%	0	100%
(5)	11g	54M	1	UDP	1470	100%	100%	0	100%
(5)	11g	54M	2	UDP	1470	100%	100%	0	100%
(5)	11g	54M	3	UDP	1470	100%	100%	0	100%
(6)	11g/11b	54M/11M	2	UDP	1470	100%	100%	0	100%

Table 10 - Test Scenarios

6.3 Test Tools

- Tshark/Wireshark [25]
- The WLAN Traffic Visualizer developed in the project (see Chapter 5).
- Iperf for the generation of data packets and logging of bandwidth [17]

6.4 Test Execution

The test equipment was connected according to the previous chapter. Two main test configurations were used, one with a single station and another with 3 stations.

The equipment list is given in Table 11.

Id	*Computer*	*WNIC*	*OS*
Sniffer	Toshiba laptop	3com 3CRPAG175B with XJACK antenna	Linux Ubuntu
STA1	Asus stationary PC	D-Link DWL-G122 USB	Linux Ubuntu
STA2	Acer laptop	D-Link G630	Windows XP
STA3	Asus laptop	CISCO AIRONET CB21AG	Windows XP
Server	Tundra stationary PC	Wire to router	Windows XP
AP	Linksys	WRT54GR	-
STA4	Toshiba laptop	D-Link DWL-G650+	Windows XP

Table 11 - Equipment list

Before running test cases, we used quite some time to find good locations for stations and the sniffer. We found that even if the distance between the access point and stations is modified a little, the sniffing result varied much. Our ideal test setup would give sniffing with minimal loss of packets.

Our tests were conducted by transmitting regular UDP packets of different size and interval uplink from the stations via WLAN to the access point and then on wire to the server. The details are given in every test case. Iperf was used for the generation the traffic on the station and reception and throughput measurement at the server. By using Iperf and UDP we are able to control the bandwidth and packets sizes and predict the channel load in advance.

We had planned to fix the data rate at different levels in test scenario (3), but our access point didn't support this. We found another access point with the support for TX rate change. The problem was then that the TX rate from the access point could be set to 54, 24 Mbps, but we were not able to set the TX rate from the station side for our uplink tests. Even though we gave up setting down the TX rate we were able to vary the packet size for the later TMT comparison.

The test with simultaneous uplink and downlink traffic in scenario (5) was cancelled, because it seems Iperf doesn't support this. Even so, scenarios (1) and (6) shows two-way traffic.

6.5 Results

Our test results are taken from log files collected with Tshark at the sniffer. That means we did all sniffing firstly according to Table 10, and we then imported the logs into WlanTV for analysis and visualization. The different results are analysed in the following sections.

Test Scenario (2)

1 station, UDP uplink, full load, decreasing packet size

Log files: {udp1, udp2, … ,udp6, udp-b1,udp-b2, …, udp-b6}

	Iperf Report		*Calculation*				*Theoretical value*		
	Packet Size (Bytes)	*Average BW (Mbps)*	*Potential throughput optimistic (Mbps)*	*Potential throughput pessimistic (Mbps)*	*Measured throughput (Mbps)*	*Payload Length (Bytes)*	*TMT (Mbps) from [6]*	*TMT (Mbps) short slot, long preamble*	*TMT (Mbps) short slot, short preamble*
11g	2300	20	23.3	21.4	21.3	1209	28.5	-	28.5
	1470	22.7	26.5	24.5	24.4	1533	31.7	-	31.7
	1000	18.2	21.6	19.8	19.7	1063	26.8	-	26.8
	500	10.9	14.3	12.8	12.7	563	18.5	-	18.5
	200	5.1	8.0	7.1	7.0	263	10.6	-	10.6
	100	2.6	5.2	4.6	4.6	163	7.1	-	7.1
11b	2300	5.7	6.3	6.3	6.1	1204	5.4	6.1	6.7
	1470	6.3	6.9	6.9	6.8	1527	6.1	6.8	7.3
	1000	5.4	6.0	5.9	5.8	1063	5.1	5.8	6.4
	500	3.6	4.3	4.3	4.2	563	3.5	4.1	4.7
	200	1.8	2.6	2.6	2.6	264	2.0	2.4	2.8
	100	1.0	1.8	1.8	1.8	163	1.3	1.6	1.9

Table 12 – Results, test scenario (2)

Test Scenario (3) **1 station, UDP uplink, decreasing load, fixed packet size** **Log files: {udp1, udp-tmt-g1, … ,udp-tmt-g5, udp-b1, udp-tmt-b1, udp-tmt-b2}**									
	Iperf Report		*Calculation*						*Theoretical value*
	BW Set (Mbps)	*Average BW (Mbps)*	*Potential throughput optimistic (Mbps)*	*Potential throughput pessimistic (Mbps)*	*Measured throughput (Mbps)*	*TBusy (%)*	*TAccess (%)*	*TFree (%)*	*TMT (Mbps), short slot, short preamble*
11g	30	22.7	27.3	25.3	25.3	60	32	8	31.7
	20	20	25.5	21.9	21.8	52	33	15	31.7
	15	14.9	25.1	16.6	16.5	40	26	34	31.7
	10	10	24.7	21.0	11.0	27	17	56	31.7
	5	5	24.5	20.8	5.5	14	9	77	31.7
	1	1	24.1	23.4	1.1	4	2	94	31.7
11b	10	6.28	7.1	7.1	7.0	87	12	1	7.3
	3	3	6.4	6.2	3.3	42	10	48	7.3
	1	1	6.3	6.1	1.1	15	4	81	7.3

Table 13 – Results, test scenario (3)

Test Scenario (4) **3 stations, UDP uplink, full load, fixed packet size** **Log files: {udp-3st-g1, udp-3st-b1}**										
		Iperf Report		*Calculation*						
		BW Set (Mbps)	*$BW_{Average}$ (Mbps)*	*STA Number*	*Potential throughput optimistic (Mbps)*	*Potential throughput pessimistic (Mbps)*	*Measured Throughput (Mbps)*	*Payload Length (Bytes)*	*T_{Busy} (%)*	*T_{Access} (%)*
11g	STA1	15	7.6	1131	31.8	31.1	30.7	1533	76	21
	STA2	15	14.6	2528						
	STA3	15	9.1	1321						
11b	STA1	3	3	2168	7.3	7.3	7.2	1532	92	7
	STA2	3	3.0	2019						
	STA3	3	1.3	761						

Table 14 – Results, test scenario (4)

6.5.1 Comparison with Wireshark

We expect a good match with Wireshark's statistics report, such as summary, WLAN conversation list, endpoint, IO graphs and WLAN traffic in Statistic Toolbar. Relevant parameters are picked up for comparison. The Wireshark comparison is made on the 11g case from test scenario (4).

-		*Wireshark*	*Test Result*
Packets Number		10000	10000
Duration Time (sec)		2.005	2.005
Average Data Rate (Mbps)		31.772	31.767
Bytes Count (bytes)		7961274	7961274
Reconstructed Bytes Count (bytes)			75117
Transactions Number from STAs to AP	STA 1	2528	2528
	STA 2	1321	1321
	STA 3	1131	1131
	STA		1
Conversation Count		5007	5007
Reconstructed Conversation Count			49

Table 15 - Parameters comparison between Wireshark and test results

As we can see in Table 15 there is a perfect match between the Wireshark figures and our calculated values. Our tool give two additional counts about reconstructed bytes and conversations that are not available in Wireshark. Lost packets missed by the sniffer are located and reconstructed, so these two values are values telling about sniffing quality.

Initially, we had a mismatch with the Wireshark values. We found that this was caused by Wireshark including the length of the Radiotap header in its calculations. This typically will add 24 bytes to every packet. As the Radiotap header is an internal header that is added by the driver for incoming packets, this cannot be included in the real throughput figures. Anyway, we have added a "Wireshark compensation" option WlanTV, and this has been activated for the Wireshark comparison in Table 15.

6.5.2 Verification of free Bandwidth Estimation

We predict that real throughput and bandwidth should climb up with the increase of bandwidth report in Iperf. Busy time and access time are expected to be higher than before. When adding two additional stations in the network, traffic load should be very busy or close to 100%.

Table 13 is the one station case with increasing traffic load. We select the load by setting the load in Iperf, and Iperf also reports the obtained bandwidth. The table shows the BW reported by Iperf, the potential throughput according to our optimistic and pessimistic method and measured throughput. Additionally we also count the percentage of the bandwidth components.

Observations for the 1-station case

- We are able to fill up the channel only for the 11b mode.

- The potential throughput according to our pessimistic method has its minimum value at 15M load, while the optimistic method predicts potential throughput around 25Mbps for all cases.

What we can see from Table 13 is that the *optimistic* method estimates the available throughput to be very stable around 25Mbit/s, while for the *pessimistic* method the estimates vary more. Especially the 15M value of 16.6 is far off from the expected value. This can be explained as follows: For the saturated channel (30Mbps load) there are no spaces between the transactions, and the channel usage is optimal. When reducing the load while keeping the packet size, Iperf increases the space between the transactions, but there is still not place for additional transactions in the open spaces. The bandwidth usage is not optimal when packets are distributed evenly. This phenomenon is evident for all the load levels from 20Mbps and down to 5Mbps. For the *optimistic* method this is no issue as it considers the sum of all open spaces as available for additional transactions. In order to compensate for this weakness of the *pessimistic* method it is possible to manually override the transaction size and access time for the fill transactions. See Section 6.5.5 for more details.

Table 14 is three stations case with full load. We set the desired bandwidth for every station to a bit above one third of full bandwidth.

Observations for the 3-station case

- We are able to fill up the channel for both modes, and there is almost no potential extra throughput
- The 3 stations get different share of the throughput.
- The access time percentage for 11b is much higher than that of 11g

6.5.3 Comparison with TMT

The reported bandwidth measurements should lie below TMT values for the different packet sizes.

Observations in relation to TMT, one station case

- As we were not able to fill up the channel in the 11g case our TMT comparison with respect to 11g is only partly relevant. Even the optimistic potential throughput prediction is not getting close to the TMT.
- For 11b the TMT comparison is useful. We are able to fill up the channel completely, and we get a throughput that lies above TMT.
- When we set the packet size higher than 1470 we get a drop in the observed packet size. This is due to IP fragmentation and is as expected.
- The packet average packet size reported by WlanTV is higher than the packet size set by Iperf. This is because Iperf packet size is the payload length of the UDP packet, while the packet size in WlanTV includes IP headers and WLAN MAC header.

In the 3-station test from test scenario (4) we actually managed to fill up the channel for 11g, and we get an average throughput that is very close to TMT, and with the pessimistic channel fill we reach 98% TMT. A single point for this observation is added to Figure 29.

For 11g the result is as expected. The values for the saturated channel lay below the TMT, even in the case we managed to fill up the channel with the 3-station test. However, for the 11b mode we measured throughput values higher than the TMT, see Table 12 and Table 13. The unexpected results

are marked red. Our first theory was that the stations were using short slot time even though the access time indicates that short slot time is not allowed. We added TMT for the short slot time in the tables, but even with this correction, we still achieved higher throughput than the TMT, so there was a need for further investigations. By looking at the 11b logs in test scenario (2), we observed that the NAV values of the unicast packets was according to the short preamble, even though the access points says that that short preamble is not allowed. We made new calculations according to short slot and short preamble, and finally we got our measured values below the TMT. It seems quite clear that STA1 is not following the directions given by the access point regarding slot time and preamble length.

The measured throughput values and TMT from Table 12 are illustrated in Figure 29.

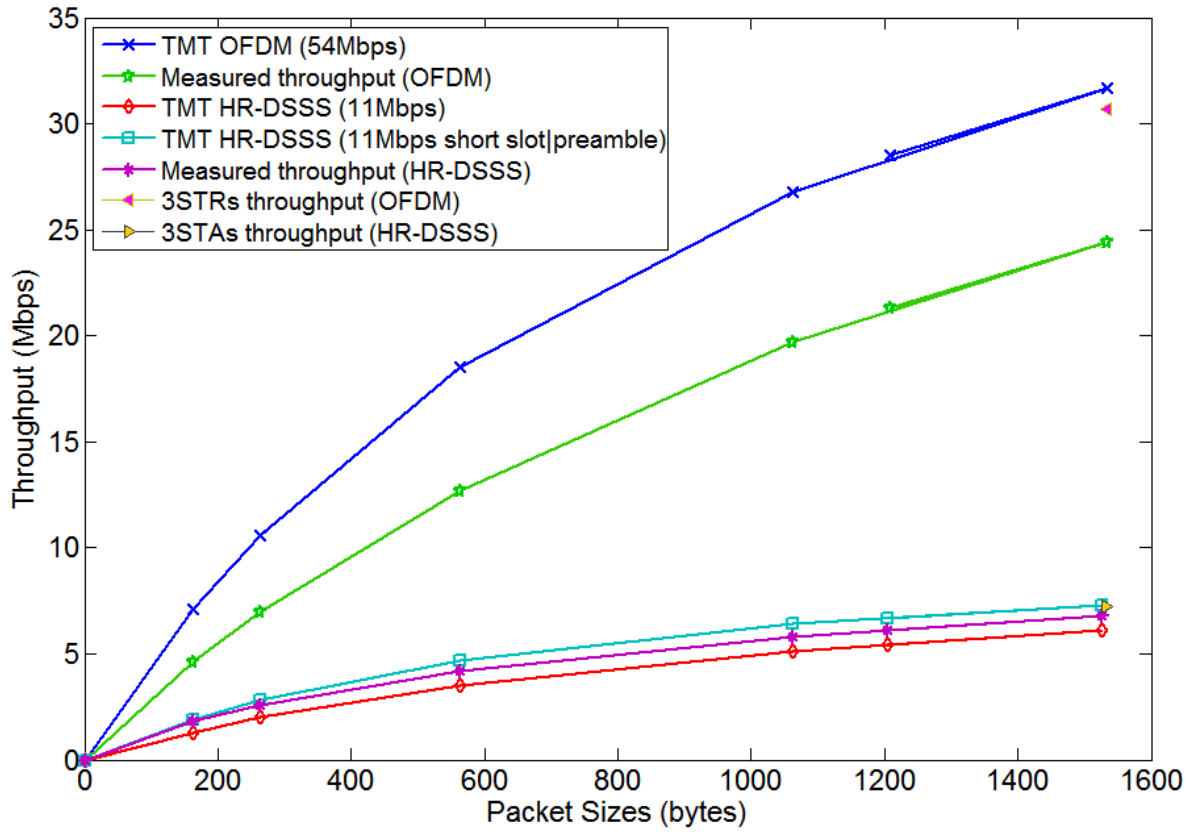


Figure 29 - Comparison between TMT and average throughput in 802.11b/g

6.5.4 Filter Function

The purpose of every filter function was introduced in Section 4.4.3. To check the filter function we check it towards the Wireshark filters. A log file with a lot of interference and more than one channel was selected. In Table 16 the result of our comparison is given. The table shows the number of remaining unicast packets after the filter was applied.

Observations

- We found that the channel filter was not working as it should.
- There are minor discrepancies that we have not investigated further.

-		*Wireshark*	*WLAN TV*
RSSI filter (-40dB)		30	32
STA filter	STA1	400	399
	STA2	81	81
	STA3	42	36
AP filter		19	21

Table 16 - Filter function test (conn-problem.cap)

6.5.5 Transaction Fill

At the higher zoom levels, the transactions added to fill up the channel are displayed in the train chart, that is, when the Fill radio button is selected. This way it is possible to check the transaction fill. The default behaviour of the transaction fill is to fill with a transaction that is similar to the logged transactions. In case this is not optimal, it is possible to adjust the access time and the transaction length, as described in Section 5.2.2.

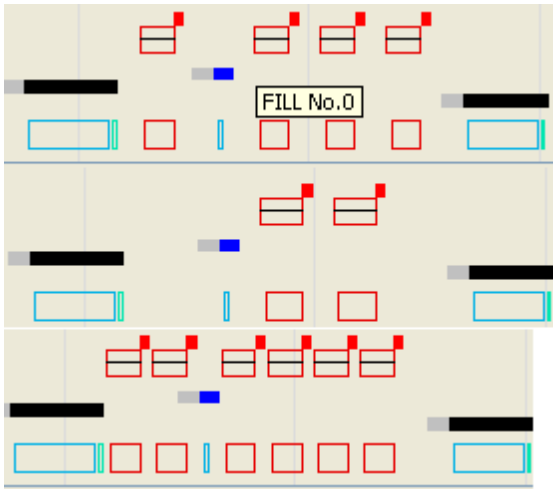


Figure 30 - Fill transaction, adjustment of access time and length

In Figure 30, the upper picture is an example of fill with automatically calculated transaction length of 803 bytes and access time of 100us. The middle picture shows the effect of changing the transaction length to 1000 bytes, and the lower picture demonstrates what happens when the access time is changed from 100us to 10us.

For this specific log the effect of changing the transaction size from default to 1470 bytes is that the number of potential extra transactions goes from 53210 to 42285. For the same log, changing the access time form default up to 350us brings the potential extra transaction count from 52310 to 38107. This will also affect the potential extra throughput.

6.5.6 Check NAV against ACK Delay Time

The intention of this test is to demonstrate monitoring function of the delay averages in the conversations panel. Normal sniffing has been done. Chapters 4 and 5 give the rationale for this functionality, so this is not repeated here. However, we should describe how to interpret these data.

The information in Figure 31 comes from 3 stations test case. Let us take one station as the example (STA 00:18:F3:32:89:6A). For the downlink unicast packets we can observe an average NAV value of 58 and an average ACK delay of 40. This is considered a suspicious deviation, and a warning is given in Figure 32. Similarly, we can compare the minimum and maximum values, as well as the standard deviation (SD). In this case there is a great SD for both the NAV and the delay, and this means that we have most probably had a channel mode change in the log.

When looking at STA 00:16:E3:5E:5B:19, we can see that the NAV values are fixed to 44. In uplink and downlink directions we can see delays ranging from 27us to 124us. These are clear indication of the time stamp jitter, and a warning is reported for the 27/44 case.

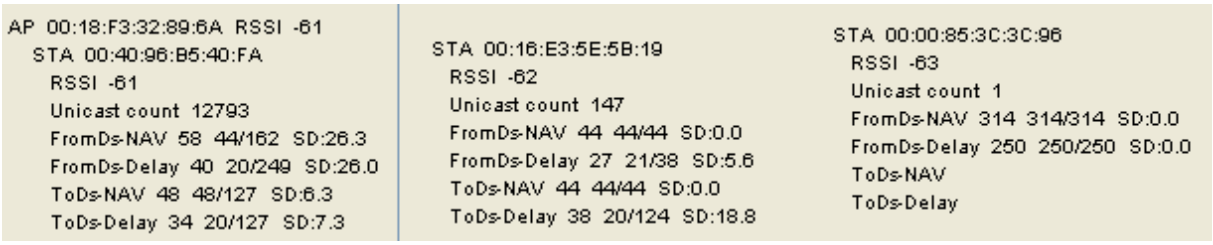


Figure 31 - Conversation report with three stations

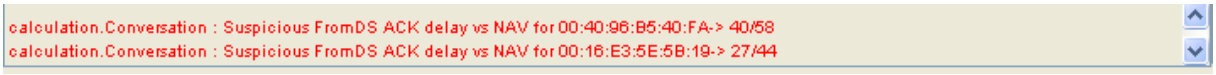


Figure 32 - Warnings in log window

We introduced the monitoring of the NAV versus ack time in order to keep track of the time stamp jitter. Figure 31 shows the conversation report for a selected log. What the figure shows is that the time stamp function is not stable. There is much jitter.

6.5.7 Mixed Mode

The mixed mode test from test scenario (6) was performed with a different test setup, with a D-Link DIR-655 and STA1 and STA4. STA4 was configured to run 11b. We obtained a throughput of 7.4 Mbps.

A clip from the test can be found in Figure 33. This figure shows a packet sent uplink in 11b mode, and then the same packet is sent downlink in 11g mode. The downlink packet is preceded with a CTS packet. Then the sequence is repeated. The upper packets are the CTS packets. One interesting observation is that the CTS as a duration (NAV) value that is far too short compared to the length of the succeeding data packet. The access point is obviously doing something wrong here, and this AP is currently test winner in the magazines! A similar case is described Figure 22, but in that case the NAV value in the CTS packet is correctly set.

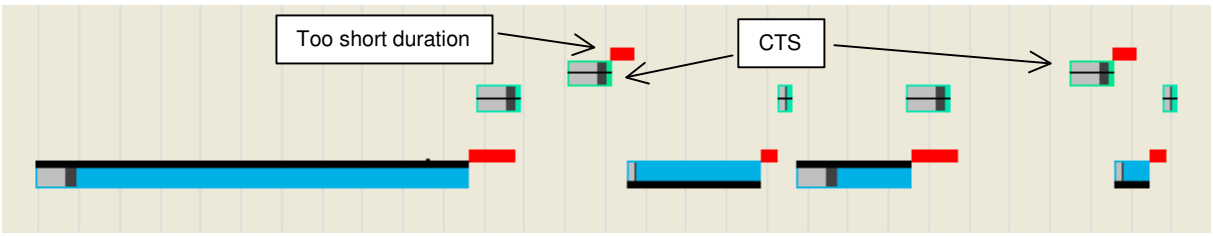


Figure 33 - Mixed mode

6.5.8 Transaction Overlap

In our final tests we still observe significant transaction overlap. Figure 34 shows overlap in 11b mode. The problem here may be that the access point announces that short slot is not allowed while the stations are actually using short slot time.

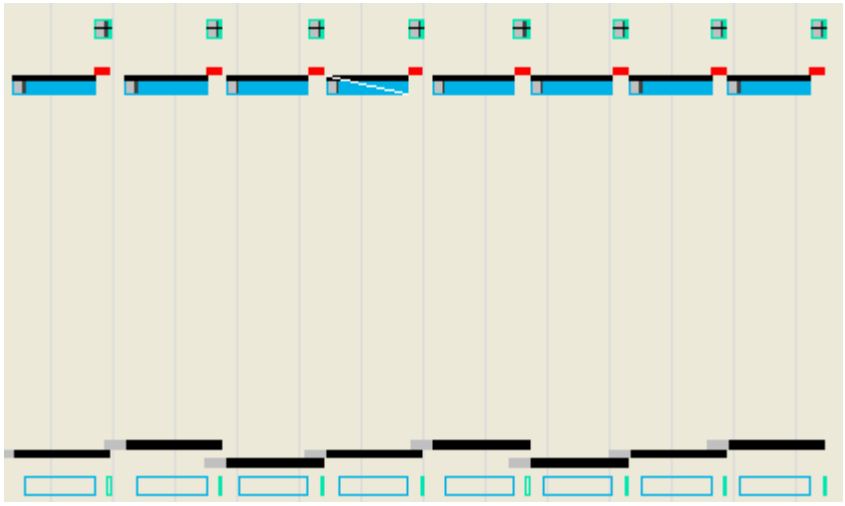


Figure 34 - Example of transaction overlap, 3 stations, 11b mode, test scenario (4)

Figure 35 shows an example of overlap in 11g mode. In this log the overlap is less evident. Even so, we have problems explaining this overlap. This is the same log giving the 98% TMT score in Section 6.5.3.

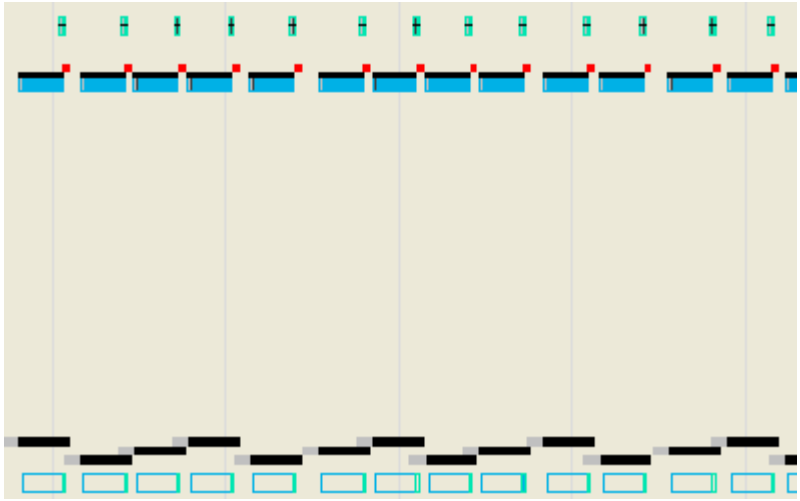


Figure 35 - Example of transaction overlap, 3 stations, 11g mode, test scenario (4)

The transaction overlap problem has already been discussed in Chapters 4.4.1.3 and 4.4.1.4.

6.5.9 TCP Traffic

Figure 36 is from log file *tcp1.cap* from test scenario (1) and it shows a typical TCP sequence with uplink payload and downlink TCP acknowledge packets. It also shows an aborted or incomplete WLAN transaction with retransmission.

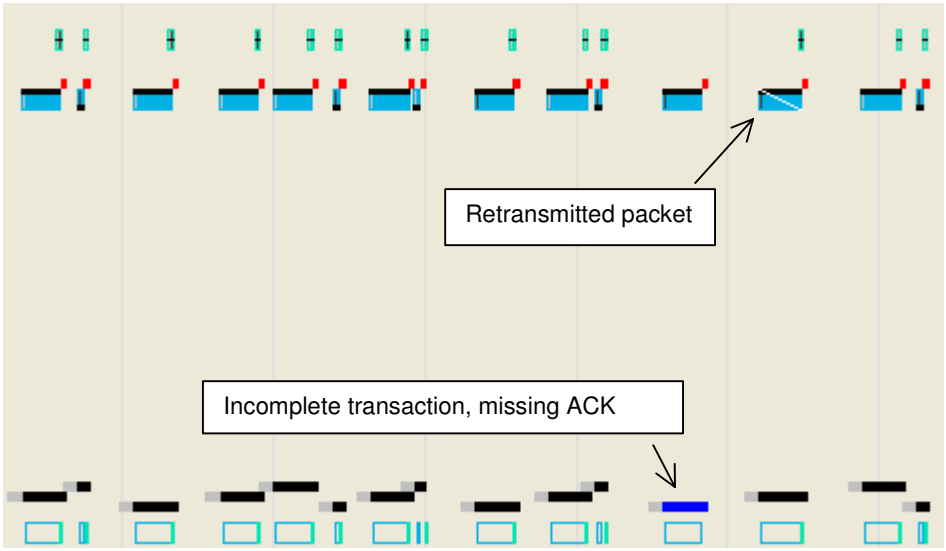


Figure 36 - TCP and incomplete transaction

7 Discussion

In this short chapter, we present discussions separately according to the four questions mentioned earlier in the problem statement.

7.1 How to Collect Data from the Radio Medium

This turned out to be a tricky part. It is easy to get data, but how to get valid data and under control is a challenge. We have been struggling with the sniffing solution, different behaviour of different WNIC and access point brands, and with the sniffer and STA locations.

Initially we tried to find tools that solved the whole sniffing and analysis task, but without success. We found tools, either very expensive, or unsupported. This led us to take the decision to make the collection and analysis tool ourselves, the WLAN Traffic Visualizer. This is a Java program running on Windows and Linux that we have developed as a part of this project.

During the tools implementation, we had problems with getting all the data we needed for the bandwidth calculations. We had to handle different header formats, headers with missing or wrong information, like wrong preamble length, and time stamp jitter. Getting the list of significant protocol fields and their interpretations right was a big task itself. When had the field list in place we had to find the right way to use Tshark for parsing the capture file, and we found a flexible and efficient solution to this problem.

Time did not allow much work on testing out sniffer locations, and we did not try out the distributed sniffing solution nor using an access point as sniffer, or sniffing inside an operating access point.

We selected the Linux sniffing solution in our final experiments as this has given the best results, even though it was not so easy to set up. Our AirPcap device on Windows is not working well anymore for the 11g sniffing – it is probably broken. That was a pity because the Windows sniffing solution was really easier to set up and more practical, requiring little manual configuration.

In some of our final experiments we lost a great part of the packets when the access point was placed close to the station, probably due to too strong signal from the access point, so that we had to move the station a few meters away. We also had problems in testing the mixed mode. It seems the access point was not compatible with the old 11b WNIC in this special test.

7.2 How to Calculate the Bandwidth Components

We ended up with using a Java program calling Tshark to parse standard capture files. With the correct WLAN protocol fields in place, the next step was to find the correct channel mode, including slot time and preamble lengths. This was more demanding than expected due to unclear documentation of header format, and different field contents with different equipment and drivers. For example, we had to get the slot time (and we should also have collected the channel) from the beacon messages.

To find the bandwidth components we first had to use a finite state machine to identify the transactions, and with the right channel mode and the length information it was fairly straightforward to find the transaction durations. The next step, finding the access time was really tricky. The access time consists of the DIFS and backoff periods, and this information is not possible to calculate or collect with the passive sniffing method. The information is simply not available on the air. Moreover,

the standard is not very clear at this point. We have not found any papers with a good solution to this problem. We ended up with a practical solution to the problem of finding the access time, where we fill up the channel with additional transactions with similar properties as the logged transactions. We implemented an *optimistic* and a *pessimistic* method for finding the potential extra load. The size of the payload of the fill transactions can selected by the user or automatically by our tool.

7.3 How to Present the Results to the Users

The main outcome of the visualization effort is the train chart. It gives the user a good overview of the whole traffic situation, and when zooming in, it is possible to see the details in message sequences and length information. It has also been extremely helpful to uncover implementation faults in our tool chain, and also to discover less good WLAN protocol implementations.

7.4 Efficiency and Precision

We have not been able to provide a complete explanation of the packet overlap and the transaction overlap problems. This can have a number of causes, like inaccurate time stamps, misleading channel modes, diverging WLAN implementations and the sniffing solution. We have seen the same in all logs, no matter what sniffing solution we have been using. However, as long as the packet sequence is right this does not have severe effect on our bandwidth component calculations in our current solution. Time has not allowed us to do a systematic analysis of this.

In the capture files we often experience that packets are lost. We have described methods to recover lost packets in order to improve the busy bandwidth calculations. Two cases have been handled; when an ACK packet is received without the corresponding unicast packet, the unicast packet is reconstructed; when a unicast packet is missing its ACK packet the NAV is used to estimate the full transaction duration. The length and access time of the added unicast packet is automatic or user configurable, in order to match the actual situation.

A basic check of the filter function was performed where we compare the filters with similar Wireshark filters. We find acceptable matches where the filter function is similar. However, our channel filter appeared to be faulty. We used the channel information in physical header, in both Wireshark and WlanTV, but this always matches the logged channel, i.e. the centre frequency of the receiver. In order to fix this, we need to get the channel from the beacon of the appropriate access point. In Wireshark such a filter is not available.

When comparing the calculation of throughput figures, packet counts, and other information with the Wireshark summary report, we initially got a discrepancy. We found that this was caused by Wireshark including bytes from the internal Radiotap header in its calculations. This gives a significant added throughput in the Wireshark reports that we consider incorrect. When adding the extra bytes to WlanTV we got an exact match with Wireshark in all comparable information. The default behaviour of WlanTV is to exclude the extra bytes.

We also compared our measured results and estimations with the TMT for the selected modes and data rates. Measured and estimated maximum throughput should lie below the TMT. Initially we got too high measured values for 11b. After closer investigations we found the explanation; the STA1 used was using short slot time and short preamble, although the access didn't allow this.

We have struggled with the interpretation of the IEEE 802.11 standard in comparison to the different vendors' realizations as observed in our logs, with respect to the channel mode information and in particular with what is called *mixed mode* in the access points, and the DIFS and backoff times.

Our limited time did not allow experiments in the more complex situations with hidden terminals or interference problems so we don't know the performance of our tool in such a context.

8 Conclusion and Future Work

In this chapter, we present conclusions and the major contributions of the thesis, as well as future work.

8.1 Concluding remarks

In this thesis, we have described and implemented a solution for data collection and bandwidth measurement in WLANs by using standard equipment and freely available tools and platforms. With the Wlan Traffic Visualizer, a tool that has been developed as a part of this thesis, it is possible to collect capture files and get a preciese overview of the traffic and load situation of the WLAN, and the bandwidth usage and potential extra throughput is clearly presented. At the higher zoom levels, the detailed WLAN packet sequences and time information can be inspected, and this can be used for educational purposes, as well as for the observation of WLAN equipment behaviour and for equipment benchmarking. We have also proposed and implemented new methods for available throughput estimation based on previous work, and through a series of experiments, our implementation has been verified towards standards and other references. As a spin-off observation directly from our visualization results, we have discovered examples of misbehaving access points, WNICs, and tools, which indicates that these implementations by different vendors are not 100% compatible with the standards.

All in all, we believe that the objectives of this Master thesis have been achieved.

As an additional note it can be mentioned that the WlanTV tool is already being used in WLAN protocol development work at EMP.

8.2 Main Contributions

The main outcome of this thesis is the WlanTV tool that we have developed for the collection, calculation and presentation of WLAN traffic and available bandwidth:

- It uses a new method for reading capture files by using the Wireshark/Tshark dissectors.
- It is running on Linux and Windows. With Windows a special AirPcap sniffer is needed for live capture.
- It is based purely on open source tools and libraries, and it can easily be extended and adapted.
- No special hardware is needed for sniffing, except a standard PC and a WNIC.

The innovative Train Chart has proved to be very useful. It can be used for different purposes:

- It gives an instant overview of the traffic and load situation
- It can be used for the verification packet sequences
- It uncovers the quality of WLAN protocol implementations
- It can be used as an instrument for the WLAN protocol training through real examples

We have developed a practical method for available throughput estimation, with two different flavors, the optimistic and the pessimistic.

We have found a number of issues with specific equipment and tools, e.g.:

- Wireshark is including the bytes of the internal Radiotap header in its bandwidth calculations.

- Some WNICs are not following the directions of the access point regarding slot time and preamble length.
- The time stamp function of the tested sniffing solutions is not stable, including the AirPcap sniffing solution for Windows.
- The preamble length information in the Radiotap header that is attached by the driver cannot be trusted.

8.3 Future Work

Our WlanTV tool has uncovered a number of issues and we have made a lot of observations that we did not manage to investigate thoroughly. We have numerous ideas about how to expand and improve the applicability of our WlanTV and the sniffing solution. Suggestions for future research are considered with two different aspects:

More effective and precise WlanTV:

- Perform measurements in more complex environments, like wireless mesh networks and distributed networks with more access points.
- Perform measurements in real operating networks.
- Add complete support for IBSS, WMM and upcoming standards like 802.11n.
- Extend the live monitoring capabilities of the WlanTV tool.
- Integration of the WlanTV with Wireshark.

More reliable sniffing:

- Investigate the time stamp function and fix its instability.
- Try out other sniffing devices on Windows and Linux to improve the capture file accuracy.
- Use the WlanTV tool to make a survey of WNICs and access points to see how they have interpreted the IEEE 802.11 standard, and in particular their behaviour in relation to access time calculations (the DIFS & backoff time implementation).
- By adding sniffing capabilities to an operative access point, it could be possible to get capture files with more complete information.

References

[1] ADHOCSYS
http://www.adhocsys.org/

[2] Bob O'Hara, Al Petrick, "IEEE 802.11 Handbook, A designer's companion," 2nd Edition. *Standard Information Network IEEE Press*, 2005

[3] Communications Network Research Institute (CNRI), Dublin Institute of Technology (DIT) WLAN Radio Resource Management
http://www.cnri.dit.ie/pres_wlan.html

[4] Frank Yong Li, Mariann Hauge, Andreas Hafslund, Øivind Kure, and Pål Spilling, "Estimating Residual Bandwidth in 802.11-based Ad Hoc Networks: An Empirical Approach," in *Proceedings of the 7th International Symposium on Wireless Personal Multimedia Communications(WPMC'04),* Abano Terme, Italy, Sept, 2004

[5] Gang Wu Fanglu Guo Tzi-cker Chiueh, "Transparent and Accurate Traffic Load Estimation for Enterprise Wireless LAN," in *Network Computing and Applications, Sixth IEEE International Symposium on Network Computing and Applications (NCA 2007),*pp.69-78, July, 2007

[6] Jangeun Jun, Pushkin Peddabachagari, Mihail Sichitiu, "Theoretical maximum throughput of IEEE 802.11 and its applications," in *Network Computing and Applications, Second IEEE International Symposium on Network Computing and Applications (NCA 2003)*, pp 249-256, April, 2003

[7] Jihwang Yeo, Moustafa Youssef, and Ashok Agrawala, "A Framework for Wireless LAN Monitoring and Its Applications," in *Proceedings of the Third ACM Workshop on Wireless Security (WiSe'04)*, pp.70-79, 2004

[8] Mark Davis, "A Wireless Traffic Probe* for Radio Resource Management and QoS Provisioning in IEEE 802.11 WLANs," in proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems (MSWiM'04), pp.234-243, Italy, 2004

[9] Martin N. Nielsen, Knut Øvsthus, and Lars Landmark, "Field trials of two 802.11 residual bandwidth estimation methods," in *IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS'06)*, pp. 702-708, Oct, 2006

[10] Mathieu Déziel, Louise Lamont, "Implementation of an IEEE 802.11 Link Available Bandwidth Algorithm to allow Cross-Layering," in *IEEE International Conference on Wireless And Mobile Computing, Networking And Communications (WiMobapos'05)*, Vol. 3, pp. 117-122, Aug, 2005

[11] Matthew S. Gast, "802.11 Wireless Networks: the Definitive Guide," 2nd Edition, *O'Reilly Media, Inc.*, Colleen Gorman, 2006

[12] Tools and Techniques for Measurement of Networks
http://nile.wpi.edu/tools/

[13] Wi-Fi CERTIFIED for WMM - Support for Multimedia Applications with Quality of Service in Wi-Fi, Wi-Fi Alliance
http://www.wi-fi.org/white_papers/whitepaper-090104-wmm

[14] www.uninett.no/wlan/

Tools and software

[15] AirPcap
http://www.cacetech.com/products/airpcap.htm

[16] Atheros
http://www.atheros.com/

[17] Iperf
http://dast.nlanr.net/Projects/Iperf/

[18] Jpcap
http://netresearch.ics.uci.edu/kfujii/jpcap/doc/
http://jpcap.sourceforge.net/

[19] Kismet
http://www.kismetwireless.net/

[20] Madwifi
http://madwifi.org/, http://madwifi.org/wiki/UserDocs/MonitorModeInterface

[21] Tcpdump/Libpcap
http://www.tcpdump.org/

[22] Tobi Oetiker's MRTG - The Multi Router Traffic Grapher
http://oss.oetiker.ch/mrtg/

[23] Ubuntu Linux
http://www.ubuntu.com/

[24] Wireless LAN Sniffer Applications and Scanners for Linux
http://tuxmobil.org/linux_wireless_sniffer.html

[25] Wireshark
http://www.Wireshark.org/

[26] Wi-Spy 2.4GHz Spectrum Analyser
http://www.metageek.net/

[27] Interference
http://h71036.www7.hp.com/hho/cache/8931-0-0-225-121.html

[28] Radio Wave Propagation
http://www.awe-communications.com/

Wikipedia

[29] http://en.wikipedia.org/wiki/Bandwidth_(computing)

[30] http://en.wikipedia.org/wiki/Distributed_Coordination_Function

[31] http://en.wikipedia.org/wiki/IEEE_802.11, http://standards.ieee.org/

[32] http://en.wikipedia.org/wiki/IEEE_802.11a-1999

[33] http://en.wikipedia.org/wiki/IEEE_802.11b-1999

[34] http://en.wikipedia.org/wiki/IEEE_802.11e

[35] http://en.wikipedia.org/wiki/IEEE_802.11g

[36] http://en.wikipedia.org/wiki/Open-source_software

[37] http://en.wikipedia.org/wiki/Throughput

[38] http://en.wikipedia.org/wiki/Wi-Fi

[39] http://en.wikipedia.org/wiki/Wireless_Multimedia_Extensions

[40] http://en.wikipedia.org/wiki/WLAN

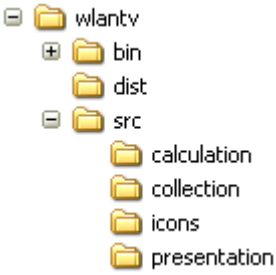
Glossary & Abbreviations

AP	Access point
DIFS	Distributed Interframe Space
EBNF	Extended Backus–Naur Form
IEEE	Institute of Electrical and Electronics Engineers
JRE	Java Runtime Environment
MAC	Media Access Control
NAV	Network Allocation Vector
OFDM	Orthogonal Frequency-Division Multiplexing
QoS	Quality of service
SNMP	Simple network management protocol
STA	Station
UML	Unified Modelling Language
VoIP	Voice over Internet protocol
WLAN	Wireless Local Area Network
WMM	Wi-Fi Multimedia
WNIC	Wireless Network Interface Card

Appendix A – Program Code and Execution

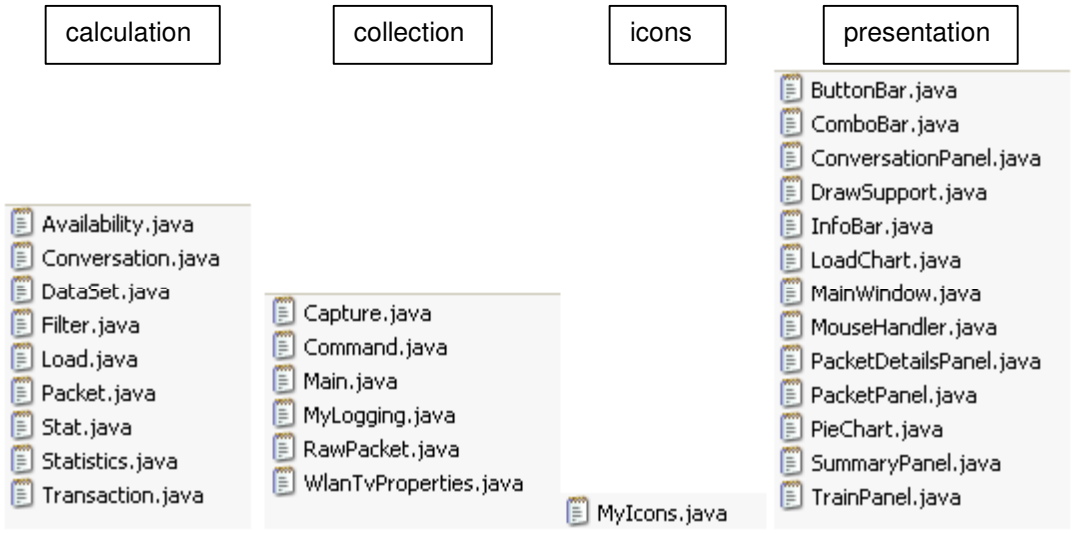
The program code can be found on the enclosed CD. It has been developed in Eclipse, and only standard libraries are needed. JRE 1.5 or later must be installed for building and running the program.

In order to build and run the program, create an Eclipse project with this structure:



A precompiled executable `wlantv.jar` can be found on the `dist` directory.

The class files are placed in directories according to package:



The icons directory also contains the icon images.

The program can be built and run inside of Eclipse. Alternatively, the program can be built by running the Ant build file called `build.xml` that can be found on the top directory. This will update `wlantv.jar` executable on the `dist` directory.

The WlanTV program uses Tshark to parse the log files. This means and `wireshark` must be installed on Windows and `tshark` and on Linux before running the program. Run by double clicking the `wlantv.jar` file if you have associated jar files with Java.

It can also be started from the command line with this command:

```
java –jar wlantv.jar
```

Appendix B – Test Results

The log files corresponding to the different test scenarios is given below, and they can be found on the enclosed CD.

Test Scenario	*Analysed in Section Number*	*Log Name*
(1)	6.5.9	tcp1.cap
(2)	6.5.3	udp1.cap, udp2.cap, udp3.cap, udp4.cap, udp5.cap, udp6.cap udp-b1.cap, udp-b2.cap, udp-b3.cap, udp-b4.cap udp-b5.cap, udp-b6.cap
(3)	6.5.2	udp1.cap, udp-tmt-g1.cap, udp-tmt-g2.cap udp-tmt-g3.cap, udp-tmt-g4.cap, udp-tmt-g5.cap udp-b1.cap, udp-tmt-b1.cap, udp-tmt-b2.cap
(4)	6.5.2	udp-3st-g1.cap, udp-3st-b1.cap
	6.5.4	conn-problem.cap
(4)	6.5.8	udp-3st-g1.cap
(6)	6.5.7	mx3.pcap