A Distributed Monitoring System for troubleshooting Wireless Networks

Ambuj Tewari
ambuj@cse.iitk.ac.in

Utkarsh Srivastava
usriv@cse.iitk.ac.in

under the supervision of

Dr. Dheeraj Sanghi
dheeraj@cse.iitk.ac.in

Indian Institute of Technology
Department of Computer Science and Engineering
Kanpur 208 016, INDIA

Abstract

Over recent years, one has seen a tremendous increase in the use of mobile devices such as laptops and PDA's replacing the conventional desktops, particularly in the realm of personal and business computing. Given such increasingly widespread use of mobile devices, one would be inclined to think that a complete switch from wired to wireless networks isn't too far in the future. However the major factor preventing this transition is that presently, the bandwidth offered by wireless networks is not even close to the maximum bandwidth achievable in wired networks.

Given the limited capacity of wireless networks due to inherent sharing of the medium, it is imperative that the available capacity be used to the maximum. Hence it is important to be able to detect and correct problems such as undue medium contention, hidden notes etc. which lead to severe performance degradation. But this requires an understanding of concepts such as a wireless channel, spectrum and the like which few, if any, network administrators can be expected to know.

Thus there is a need for an automated system which can monitor the network and automatically diagnose and correct such problems. Here we propose the design and implementation of such a troubleshooting system for wireless networks which is based on the idea of distributed network monitoring.

1 Introduction

The market for wireless communication has been growing ever since its inception. Having achieved tremendous success through wireless telephony and messaging services, it is hardly surprising that wireless communication is now beginning to be applied extensively to computing as well - the major area being that of personal and business computing. Consequently there has been a tremendous increase in the number of mobile computing devices such as laptops and PDA's. Given so many mobile devices around, a complete switch from wired to wireless networks is inevitable.

Consider the fact that 100 Mbps ethernets are now common and Gigabit ethernets also exist, while the maximum bandwidth offered by any wireless network in sight in the near future is only 54 Mbps (by IEEE 802.11a using OFDM). Thus it is apparent that wireless networks offer very limited capacity compared to wired networks and hence it is important to be able to realize this capacity as actual achieved throughput.

We look at the above goal with respect to IEEE 802.11 wireless networks operating in Basic Service Set Configuration (Section 2). It is observed in practice as well as experiment that the network seldom achieves anything close to its promised 11Mbps throughput. This may be due to several problems such as poor Access Point (AP) placement, inappropriate channel selection, presence of hidden nodes etc. Most of these problems have to do with the details of the

802.11 physical layer such as channel width, carrier sense etc.

To understand, diagnose and correct these problems, an understanding of wireless communication principles is required which can't be expected of many network administrators. It has been seen that in many commercial offices where these networks have been deployed for their sheer convenience and without adequate research, they show terrible performance. This is often because the AP's come with a default channel setting and are deployed without changing this setting. Consequently the entire network ends up sharing the same medium and the transmission is serialized leading to very poor throughput. To make matters worse, a common misconception is that the poor performance is due to inadequate number of AP's. Hence more AP's are installed leading to even higher contention and further decrease in throughput.

Thus it is clear that there is a need for an automated system which can detect such problems in the network and suggest corrective measures to be taken. Here we propose the design and implementation of such a system. The proposed framework is based on the idea of distributed network monitoring. In this approach, a number of probes are placed in the network at different locations. These probes sample the network and collect a trace of packets seen. Cross-correlation between these traces is then used in order to make inferences about the state of the network and hence about the source of the problem. Once the problem source is identified, the appropriate solution is also suggested by the system.

The rest of the report is organized as follows: In Section 2, we present an overview of IEEE 802.11 which is required in order to be able to appreciate the kind of problems that occur in its operation. In Section 3, we present experimental results demonstrating the typical poor performance of 802.11 networks which motivate the design of an automated troubleshooting system. In Section 4, the basic design of the system is presented followed by a discussion of trace-analysis techniques in Section 5. Possible system enhancements are described in Section 6 followed by conclusions in Section 7.

2 IEEE 802.11 overview

An 802.11 LAN [1] is based on a cellular architecture. Each cell, called Basic Service Set (BSS) in the IEEE 802.11 terminology, is controlled by a Base Station, called Access Point (AP). A wireless LAN can also operate in an ad-hoc mode and the presence of an AP is not required in such a case. However, it is envisaged that most installations will be formed by several BSS's where the AP's are connected through some kind of backbone, termed as a Distribution System (DS) in the standard terminology. This backbone is usually Ethernet.

All the Basic Service Sets, along with their Access Points and Distribution System, are seen as a single network by the layers above the MAC layer. This combined network has been given the name Extended Service Set (ESS). The concept of a Portal is also defined in the standard, whose function is to act as a bridge between an 802.11 and another 802.x LAN. More often than not, the Access Point and Portal reside on the same physical entity.

As is true for all 802.x protocols, the 802.11 protocol covers the MAC and physical layers. The standard defines the interaction of the MAC layer with three different kinds of physical layers:

- Frequency Hopping Spread Spectrum (FHSS)
- Direct Sequence Spread Spectrum (DSSS)
- Infra-Red (IR)

Below we give a brief description of the physical and MAC layer specifications of the 802.11 standard. We will focus only on issues which of relevance to us. Therefore, we will not go into the details of the FHSS and IR physical layers. We will also not discuss any security issues in detail.

2.1 The DSSS Physical Layer

The most popular PHY layer is DSSS. It uses special modulation techniques to spread the signal on a larger band to minimize localized interference and background noise. The original specification had provided 1 Mbps and 2 Mbps data communication capacities, but later, data rates of 5.5 Mbps and 11 Mbps were also supported.

The standard defines 11 channels on which 802.11 devices can operate. The center frequencies of these channels vary from 2412 MHz (for channel 1) to 2462 MHz (for channel 11). Since the signal is spread over a large band (about 22 MHz), transmissions on nearby channel cause mutual interference. The standard mentions that in a multiple cell network topology, overlapping and/or adjacent cells using different channels can operate simultaneously without interference if the distance between the center frequencies

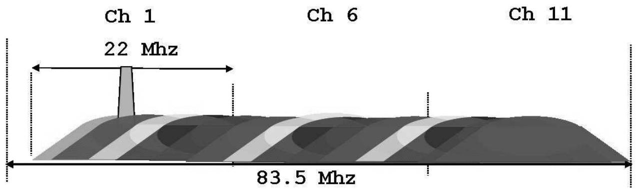

Figure 1. Channels in the DSSS PHY layer

is at least 30 MHz. In practice, channels 1, 6 and 11 provide us with three mutually non-overlapping channels because they are separated by 25 MHz and a channel is only 22 MHz wide (Fig. 1).

2.2 The MAC layer

The basic access mechanism, known as the Distributed Coordination Function, is basically a Carrier Sense Multiple Access with Collision Avoidance mechanism (CSMA/CA). There is provision for another access mechanism, called Point Coordination Function, which is meant to be used for implementation of time-bound services like voice or video transmission. In a CSMA protocol, a station first senses the carrier for ongoing transmission and delays its transmission if the medium is found to be busy. Collision detection schemes, like Ethernet, rely on the ability of the stations to detect collisions and subsequently require them to go into a retransmission phase. Collision detection is not suitable for wireless LANs for two reasons. First, it requires use of an expensive radio that is capable of transmitting and receiving at the same time. Further, it assumes that all stations can hear each other, an assumption not valid in case of a wireless LAN.

The collision avoidance mechanism used in IEEE 802.11 LANs is as follows:

1. A station that wants to transmit senses the medium. If the medium is busy then it defers. If the medium is free for a specified amount of time (called Distributed Inter Frame Space (DIFS) in the standard), then the station is allowed to transmit.

2. The receiving station verifies the integrity of the received packet by calculating the CRC of the received packet and sends an acknowledgment back to the transmitter. If the sender does not receive an acknowledgment, it keeps retransmitting until it receives one or the limit for the maximum number of data retries is exceeded.

2.2.1 Virtual Carrier Sense (RTS/CTS)

To further reduce the probability of two stations colliding because they cannot hear each other (such nodes are called "hidden nodes"), a mechanism called Virtual Carrier Sense, is defined. When a station wishes to transmit a packet a short control packet called RTS (Request To Send), which includes the source, destination, and the duration of the following transaction (packet + the respective ACK), is sent. The destination responds to this packet if the medium is free and sends a CTS (Clear To Send), which includes the same information. All stations receiving either the RTS or the CTS, set their virtual carrier sense indicator (called Network Allocation Vector (NAV)), for the given duration, and use this information together with actual carrier sense when sensing the medium. If the overhead involved in detecting a collision is significantly larger than the overhead required to send those extra RTS/CTS packets, then using the virtual carrier sense mechanism is a benefit. The standard defines an RTS threshold and all packets smaller than this value are transmitted without the RTS/CTS transaction.

2.2.2 Fragmentation and Reassembly

Other LAN protocols use packets which are several hundred bytes long. In the wireless case, however, several reasons make it preferable to use smaller packets. Firstly, due to higher error rate of a wireless link, the probability of corruption of a packet increases with its length. Secondly, smaller packets cause less overhead on retransmission.

Not allowing larger packets in wireless LANs is not a good option because we will not be able to deal with Ethernet packets. Therefore, there is provision for fragmentation and reassembly at the MAC layer itself. The mechanism is very simple to understand. The transmitting station after sending one fragment is not allowed to send another until:

1. It receives an ACK for the said fragment, or

2. Decides that the fragment has been retransmitted more than a specified limit and drops the whole frame.

2.2.3 Inter Frame Spaces

In the standard, four types of Inter Frame Spaces are defined so that different priorities may be provided

The Short Inter Frame Space (SIFS) is used to separate transmissions belonging to a single dialog (e.g. a fragment-ack), and is the minimum Inter Frame Space. This provides the highest priority and its values depends on what the underlying PHY layer is.

The Point Coordination Inter Frame Space (PIFS) is used by the Access Point, to gain medium access before any other station. The value is SIFS plus one Slot Time, which is the unit for exponential backoff as described in the next section.

The Distributed Inter Frame Space (DIFS) is used by a station that wishes to transmit a new packet. It is PIFS plus one Slot Time.

The Extended Inter Frame Space (EIFS) is the longest IFS and is used by a station when it receives a packet which it does not understand.

2.2.4 Exponential Backoff

Backoff is a mechanism to resolve contention for accessing a medium. The method requires each station to choose a random number between 0 an n, and wait for that many slots before again sensing the medium for retransmission. The slot time is defined in such a way that a station can determine if another station has accessed the medium at the beginning of the previous slot. Exponential Backoff means that the maximum random number n is increased exponentially whenever collision is detected. The standard specifies that the exponential backoff algorithm be used always except when a station is sending a new packet and has seen the medium free for more than DIFS.

2.2.5 Joining a BSS

If a station wishes to access an existing BSS, it needs to get certain synchronization information from the AP. It can do this via two alternate means:

1. Passive Scanning: The station just waits to receive a beacon frame, which is a frame sent periodically by the AP and it contains synchronization information.

2. Active Scanning: The station scans for an AP using a Probe Request Frame and waits for a Probe Response to arrive.

The wireless card firmware automatically detects when the signal is deteriorating and it needs to execute a handoff. Within a single BSS too, the data-rate of transmission is decided on the basis of the signal quality. Data rates ranging from 1Mbps to 11Mbps may be chosen.

The standard also proposes security mechanism which require authentication before association. The authentication can be based on the possession of a secret key. Eavesdropping is prevented by encrypting the transmitted packet using a pseudo-random sequence derived from a shared secret key.

3 Network Problems and Causes

IEEE 802.11 can operate at a maximum data transmission rate of 11 Mbps. We first motivate the design of an automated troubleshooting system by presenting experimental results where the actual throughput observed is nowhere close to the above figure. All the throughput measurements reported below were made with the network benchmarking tool *netperf* [3] and for a TCP connection.

3.1 An ideal scenario

We measured the throughput obtained with a single node operating alone in its BSS and close to the AP while the other BSS were operating in non-overlapping channels. AP's A, B, and C (Fig. 2) were operating in channels 1, 6 and 11 respectively. A single wireless node was placed at P3 and associated with A. Thus there was effectively no contention for the medium. A TCP connection was set up between the wireless node to another node on the wired side. However, even under these ideal conditions, the throughput observed was only 5.3Mbps.

This can also be explained analytically. First we assume that the node is at a sufficiently small distance from the AP for the data rate of 11Mbps to be achieved. We also neglect the overhead of control and management packets.

The TCP send process by the node is as shown in Fig. 3.

In case of DSSS, SIFS = 10 μs, DIFS = $50\mu s$, Contention Window = 32 and Slot time = $20\mu s$. The average backoff is given by

$$(Contention\_Window)/2 * Slot\_time = 320\mu s$$

Physical Layer Convergence Procedure (PLCP) is completed at 1Mbps while the other parts are transmitted at 11Mbps. The total time for the send process is thus easily seen to be $1900\mu s$.

The TCP ack process is as shown in Fig. 4. The total time for the ack process can be calculated as $846.9\mu s$. Since TCP implements delayed acks, a cycle consists of

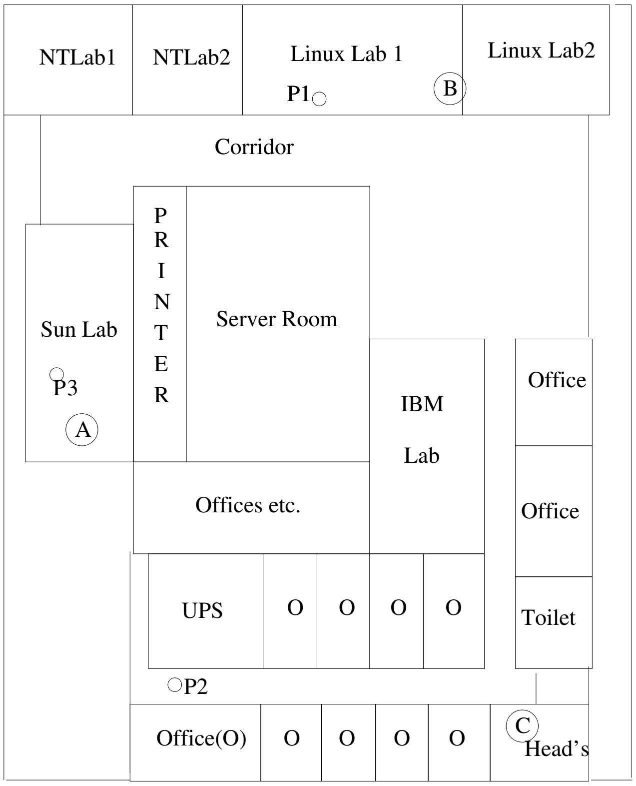

Figure 2. Experimental Setup

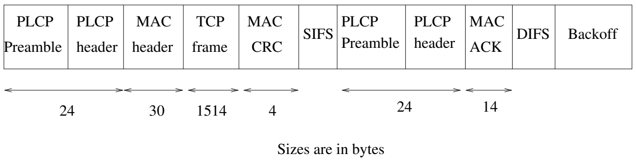

Figure 3. TCP Send process

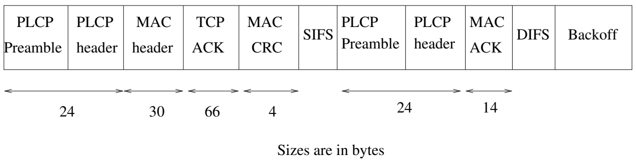

Figure 4. TCP Ack process

send, send and then ack. Hence a total of 3094 bytes are transmitted in 4646.9μs. Thus the throughput is 5.33Mbps.

Thus it can be seen that the actual achievable throughput agrees closely with that observed experimentally. This is because the network in this case is a healthy one and there are no problems such as hidden nodes, medium contention etc. However this example is relevant because it demonstrates that even under ideal conditions the achievable throughput is far below the maximum achievable data rate. Furthermore, if any problems exist in the network, the throughput falls drastically as shown in the next scenario.

3.2 A network with hidden nodes

In this case two nodes (X and Y) were placed at positions P1 and P2 and associated with A. B and C were operating on non-overlapping channels. It was tested that the nodes were hidden from each other by operating the network in ad-hoc mode and ensuring that they couldn't ping each other. Throughput measurements were made by setting up TCP connections between X and Y and also from X to Z (a node on the wired side) and Y to Z. The results are summarized in Table 1.

It can be seen that even with a few nodes network performance can undergo severe degradation. Moreover, unexpected problems can crop up such as a skew in throughput observed by X as compared to that by Y. It maybe hypothesized that X is located at a more advantageous position. But the experiment was repeated with X placed at P2 and Y placed at P1. The results are summarized in Table 2.

It is seen that X continued to show a higher throughput inspite of its changed location. As we shall see in Section 5, this can be explained on the basis of different receiver sensitivity or different firmware versions of different cards.

3.3 Causes of poor performance

The above example just demonstrates poor performance in case of one of the several problems possible in a wireless network. However, after careful thought process and experimentation, we can see that there are various possible causes of poor performance. These can be enlisted as follows:

1. Presence of hidden nodes

2. Poor Channel Selection

3. Power setting of AP causing undue interference

Scenario	Throughput at X (Mbps)	Throughput at Y (Mbps)
X to Y, no RTS	0.38	0.25
X to Y, RTS=1000	0.66	0.15
X,Y to Z, no RTS	4.10	0.57
X,Y to Z, RTS=1000	4.15	0.26

Table 1. Observed throughput with hidden nodes

Scenario	Throughput at X (Mbps)	Throughput at Y (Mbps)
X,Y to Z, no RTS	3.82	0.1
X,Y to Z, RTS=1000	2.68	0.51

Table 2. Observed throughput with the positions of X and Y swapped

4. Coverage problems due to poor AP placement
5. Presence of sources of interference (microwave etc.)
6. Skew in observed throughput between two cards due to difference in sensitivities
7. Inappropriate algorithms used for handoff causing too many oscillations in association.

Clearly, manual diagnosis of these problems is going to be hard. Hence there is a need for an automated system which can diagnose and correct these problems.

4 Design of the System

Our monitoring tool analyzes the state of the network and suggests possible causes of poor performance using a distributed set of probes which are placed at appropriate locations in the network. This methodology is similar to the one performed for analysis of TCP packet dynamics across the Internet in [2]. A probe is nothing but a laptop equipped with a wireless card and running a trace uploading utility. There is a central server to which these probes upload the traces. This is done using a trace collection utility described in Section 4.1 below. Later in this section, we also examine the issues involved in placement and selection of channel scanning mode of the probes. The probes upload the data independently of each other and the data is merged into a single database on the server side. We do not require the clocks of these probes to be synchronized with each other.

4.1 Trace Collection Utility

To analyze the events happening in a wireless network, we require access to a database containing a record of the packets that were transmitted over the wireless LAN. There are softwares available which enable us to log information about each packet received by a wireless card. Such sniffers typically offer a GUI interface to setup custom filters using which one can can log only those packets which satisfy a given condition. The sniffer we used could operate in two modes: 1) either fixed on a given channel, or 2) scanning several channels, staying on each of them for a specified period of time.

It is easy to obtain a single trace of packets seen by the wireless card using such a sniffer, but it is not convenient to use it for automatic data collection. Setting configuration parameters in the sniffer and uploading the obtained traces to a central server requires manual intervention. We automated the entire trace collection task by building a set of client-side and server-side utilities. This was done in collaboration with two more colleagues.

4.1.1 Architecture

The basic approach of the trace collection utility is a client-server one. A typical trace collection experiment consist of placing several laptops, each running the client-side utility, at various points in the wireless LAN. The client invokes the sniffer and uses the POST mechanism of the HTTP protocol to upload the traces to a central server (Fig. 5). On the server side, the traces are inserted into a PostgreSQL

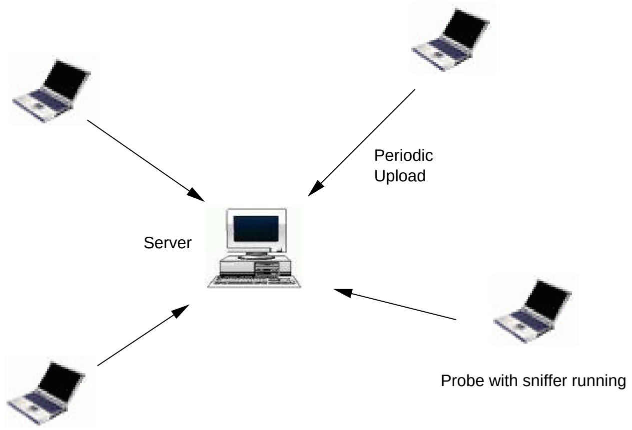

Figure 5. Architecture of the trace collection utility

database. Using a database makes the post-processing of these traces more convenient since tuples of interest can be extracted by simple SQL queries.

4.1.2 Implementation

The client utility was implemented in Visual Basic. It invokes the sniffer and by sending keystrokes to the GUI application it automatically starts the trace collection process. While the sniffer is running, the network card can no longer be used for normal wireless communication. Therefore, the utility stops the sniffer and POSTs the traces to an HTTP server. After that, it again starts the sniffer.

A packet has a several pieces of information associated with it, e.g. source, destination, signal strength, channel on which received, etc. The client uploads only a user specified subset of these. This is important, since we do not want to flood the network with data generated by our utility. Only relevant data should be uploaded to the server. In the current implementation, the client does not apply any pre-processing on the traces. Information about all packets seen, even about those with CRC errors, is uploaded.

The server side utility is a CGI script written in Perl. It connects to a PostgreSQL server running on the local machine and inserts the received tuples into a database. It also sends back an error code back to the client in case of an error. The script is written in Perl and so can be ported to different systems without significant changes. Since the uploading mechanism is HTTP and we store the traces in a PostgreSQL database, the utility can be easily adapted to work on a variety of systems. Also, the client and server side programs are almost independent of each other.

4.2 System setup

Before starting the process of trace collection, a number of issues need to be taken care of, so that we obtain maximum information from the analysis of these traces. Such issues include:

- Positioning of the probes
- Number of probes required
- Channel sampling mode

The right positioning of probes is very important if we want to derive useful information about the state of the network from the traces collected. Placing a probe near each AP in the network is recommended since that probe gets to receive all the packets sent to and received by the AP. Also, this is required for the analysis of hand-offs as described in Section 5 and can also be used for building a location map as in Section 6.

Apart from placing a probe near each AP, some more probes should be placed distributed well over the wireless network. In general, more probes are needed if there are many hidden nodes in the network. If too few probes are used, there is a possibility that some hidden nodes will not be detected.

The probe should listen on a fixed channel instead of cycling through all the channels because of the following two reasons. First, it will not be possible to do packet-correlation if probes are cycling through the channels. This is because the probes are not synchronized with each other. Second, we do not really need to observe activity on all channels. We just need to fix the probe to that channel to which a node would have tuned itself, had it been in that location. This is because if we listen to some other channel and detect problems on it, we do not detect anything useful since a node at that location would not be tuned to that channel. A node tunes itself to the channel on which the signal is strongest. Therefore, a probe should listen on a channel where the signal is strongest.

5 Trace Analysis Techniques

Once the traces have been uploaded to the central server, they have to be analysed to identify any problems in the network. Almost all the problems enlisted in Section 3.3 can be diagnosed by an analysis of the appropriate kind of packets from the traces and cross-correlation between the packets uploaded from different probes. We carried out this analysis on traces collected from a busy wireless network having 7 access points and about 20 nodes. The access-points were operating in non-overlapping channels 1,6 or 11.

5.1 Pre-Processing

Before proceeding for detailed problem diagnosis, some preliminary processing on the traces is required. Specifically, we can deduce the following:

1. *Visibility Graph*: We can build a visibility graph at each probe, indicating which nodes are visible at that probe. This can be built by recording the last time at which the probe saw a packet from the node. After a certain timeout has occurred, the node is assumed to have become invisible from that probe.

2. *AP channels*: The AP's periodically broadcast beacon packets which contain information about the channel on which they are operating.

3. *Node associations*: Since each packet is transmitted from a node to an AP or the other way, it is always possible to deduce the AP with which each node is associated. Furthermore, it is possible to detect handoffs through dissociation and reassociation packets.

4. *Channel at which each packet was transmitted*: Each packet is either transmitted to or from an AP and hence contains its MAC address. Since the AP channels are known, the channel of the packet can be deduced. The ACK packets form an important class of exceptions. They contain only the receiver address which may not be an AP. But this is not a problem since the node associations are known and a node operates on the same channel as the AP with which it is associated.

5. *Time Synchronization*: In general it is not essential that the clocks of the different probes shall be synchronized. In this case, the state of a network at the same instant will be reported with different timestamps from different probes. This makes cross-correlation between the traces from different probes difficult. Cross-correlation is especially required for the detection of hidden nodes as described in the following section. To make this step easier, we perform a time synchronization between the different probes. This can be done by looking for the same beacon packet (as identified by their sequence number) in traces uploaded from different probes. This can be used to deduce the clock offsets between the various probes. Once the clock offsets are known, the corresponding correction can be applied to each trace, to obtain the network view at the same instant from different probes.

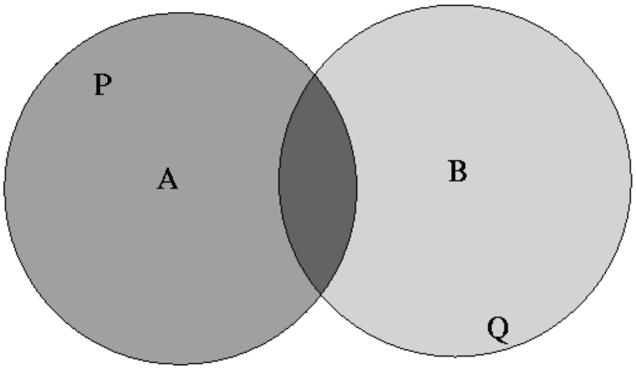

Figure 6. Detecting hidden nodes
A,B: Nodes P,Q: Probes

5.2 Finding Hidden Nodes

Distributed trace collection can also help in finding out the hidden nodes present in a network. Two nodes are hidden from each other if neither can hear the transmission of the other, but there is a part of the network which is in the range of both.

Our strategy for identifying hidden nodes is as follows. If for certain nodes A and B, we see a packet transmitted by A at a probe, say P, and at the same time we see a packet transmitted by B, on a channel overlapping with that of A, at a another probe, say Q, then we say that nodes A and B might be hidden from each other. (Fig. 6)

To be certain that A and B are hidden, all we need is the existence of a probe from which both A and B are visible. In case A and B are in the same cell, there is guaranteed to be such a probe, the one that is placed at the AP. Otherwise, we have to consult the visibility graphs at the various probes which is described in Section 5.1.

It is clear that if we declare A and B to be hidden then they are actually so. Simultaneous reception of a packet transmitted by A and a packet transmitted by B at two different probes is evidence of the fact that A and B cannot hear each other or else they wouldn't have transmitted simultaneously. Presence of a third node from which both are visible, guarantees that A and B's ranges overlap. Thus, there are never false alarms about hidden nodes. However, hidden nodes may go undetected if there are not sufficient probes to detect simultaneous transmission or if there is no probe in the region of overlap of the two hidden nodes' ranges. This problem can be handled by using more number of probes.

It is essential that this step of finding hidden nodes be carried out after time synchronization, as described in Section 5.1, has been done. This is because we need to find out an occurrence of *simultaneous* transmission which is not possible unless we have synchronized the time of the packets collected at different probes.

5.3 Analysing Checksum Errors

It was observed that the collected traces contained a very large number of checksum errors. In our case, this was partially due to the fact that the probes were not sampling the network in accordance with the guidelines prescribed in Section 4.2. For instance if a probe is placed in a cell operating in channel 1, and it is listening on channel 5, it will practically observe all the packets in error because the signal to noise ratio on that channel will be very low, since the amount of overlap is very little.

Nevertheless, when the probes are listening to the appropriate channel, the packets in checksum error are those which carry all the information about the problems occurring in the network. Hence these packets need to be carefully analysed.

However a major difficulty that comes up while analysing these packets is that the address field of these packets is often garbled, thus making it impossible to determine from which node or on which channel the packet was transmitted. However, a large number of such error packets are immediately followed by an ACK in the traces. This is because the ACK is transmitted at a bit rate of 2Mbps and hence can be captured at a lower SNR than the original packet. The receiver field in the ACK then reveals the channel at which the original packet was transmitted. The percentage of CRC errors in our traces on different channels before and after the above correction are shown in Fig. 7. It can be seen that the CRC errors are high on channels other than the ones on which the AP's are operating.

Checksum errors may occur due to:

- Collision at the probe even though high SNR

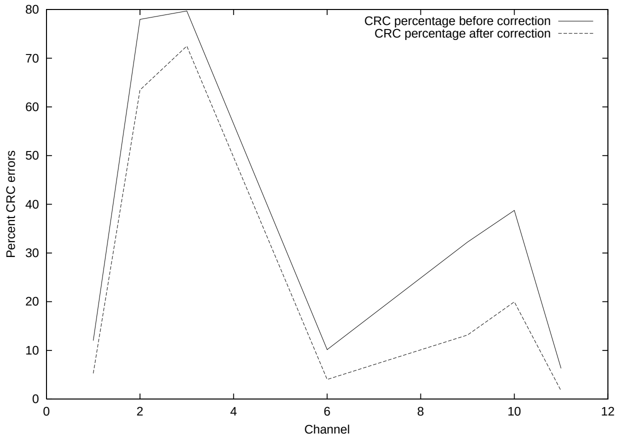

Figure 7. Percentage of CRC errors

- Low SNR at the probe

Both the above factors have different implications related to the kind of problems that may be occurring in the network. The decision tree which is to be followed to deduce the problem from checksum errors is shown in Fig. 8.

When a CRC error is observed at probe P at time t, it can be either with high SNR in which case it points towards a collision or maybe due to low SNR. If it is due to a collision, it is probable that hidden nodes will be detected through some other probes at the same time. In this case, the use of RTS/CTS is advisable. If hidden nodes are not detected and CRC errors with high SNR occurs repeatedly, it points towards the presence of a source of interference such as a microwave oven.

On the other hand, if the CRC error is due to low SNR and it is observed that the percent of CRC errors at P is very high (of the order of 90% or more), it indicates that the probe is not within the coverage area of any BSS. This coverage problem can be solved by bringing the AP closer to P or by increasing the power, or by installing a new AP.

If the percent of CRC errors with low SNR is moderate and the packets with CRC errors are from an AP (B) different from the AP (A) to which P is associated, it indicates that the packets from another cell are leaking into this cell. This is disadvantageous because if a node would have been present at P's position, it would have sensed this packet and not transmitted, leading to serialization of transmission between the two cells. This can be remedied by either changing B's channel to be non-overlapping with A or by decreasing the power of B.

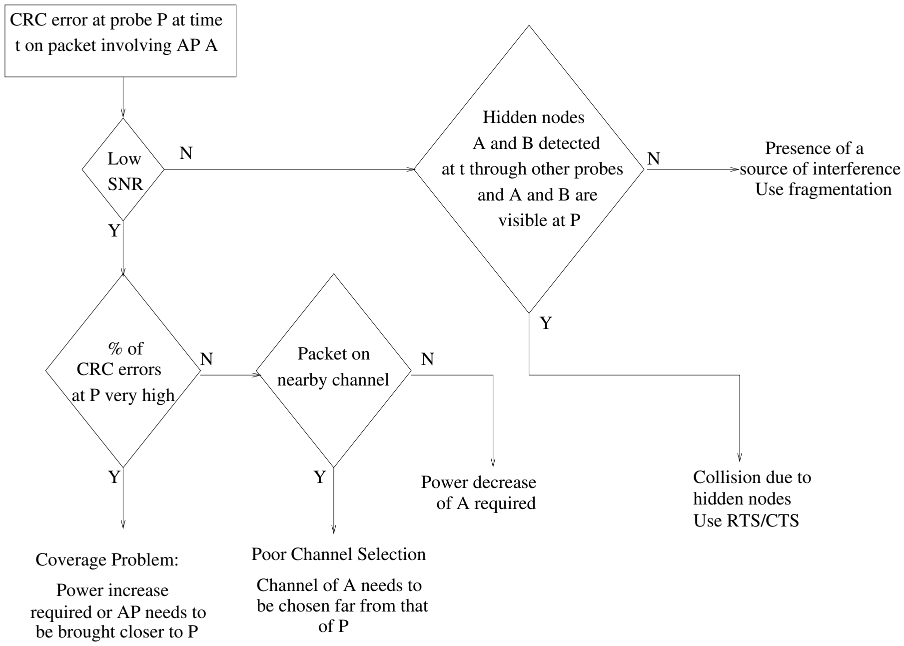

Figure 8. Decision tree for checksum errors

Each solution has its disadvantages. Decreasing the power may begin to cause coverage problems. Changing the channel may cause the AP to interfere with another BSS. Thus each solution can be tried in turn, retaining the one which causes best network performance.

5.4 Analysing Handoffs

It was observed in the traces that a node often oscillated frequently between two AP's, dissociating from one and then reassociating with it in a matter of seconds. Such excessive oscillation can lead to poor network performance. Assuming that the user is not moving with such a speed, this clearly points to a deficiency in the handoff detection algorithm followed by the card firmware, since the node associations are happening on the basis of temporal rather than spatial variation.

This problem can be detected and the deficiencies in the handoff detection algorithm can be analysed with the help of the probes placed at the AP's. Assuming symmetric propagation, the signal strengths observed at the AP's involved in the handoff from the node executing the handoff shall follow the same pattern as the signal observed at the node from the AP's. Clearly, there is a problem if a node N switches from AP A to AP B and the signal strength from N received at B has not been increasing substantially or that at A has not been decreasing.

5.5 Wireless Card Behaviour

It is often observed that network cards from different vendors perform differently under the same network conditions. Even with the same vendor, a card with a different firmware version might not perform in the same way. There are a couple of reasons why this happens. First, a card that has a very sensitive receiver will detect the medium to be busy even when the other would have sensed it to be free. It will, therefore, backoff and not transmit. Second, a card might employ a very aggressive contention window strategy and not expand its contention window in accordance with the standard. This will provide the card more than its fair share of network bandwidth.

To detect such a phenomenon in a set of cards, one can do the following experiment. The cards are used to flood the network with traffic and probes are used to record the traces. We can then calculate the inter-arrival times for packets transmitted by each node in the network from the traces. A node which shows a low value of inter-arrival time consistently has either a less sensitive receiver or is employing an aggressive contention-window strategy.

6 System Enhancements

6.1 Real Time Version of the System

The current system that we have built is an off-line system. Traces are collected and analyzed offline using a set of scripts written especially for the purpose. The scripts generally scan the entire trace to generate various outputs. Ultimately, it is desirable to develop a network monitoring system working in real-time.

In such a scenario, the clients will periodically upload the data as before but the server will not insert the data into a database. Instead it will maintain summary information needed to compute the final output. For example, to calculate the final percentage of packets which were affected by CRC errors, one only needs to maintain the number of packets, with and without CRC, seen so far. It is not required to store all the packets in a database.

As another example, for maintaining visibility information, we will need to store the last time a node saw a packet of another node. Also, we will not know the number of AP's in the network beforehand and thus we will have to incrementally construct the set of AP's as we see more and more packets. It is possible to adapt each of the currently used algorithms to a real-time environment.

6.2 Client Side Filters

Presently, there is a unidirectional flow of information from clients to the server. This results in massive uploads from the client side because it does not receive any instructions from the server as to what information it can discard on the spot. This is a problem because we are somewhat aggravating the problem we are trying to solve, that of capacity wastage. A desirable feature would be to have the server tell the clients what information it wants to store in the database and the clients can then apply filters to the data before uploading them to the server. These filters can change over a period of time and the server can periodically instruct the clients to make the changes.

For example, if one is interested only in handoff analysis then only the probes near AP's need to upload. In the same way, if only time synchronization is to be performed, the server can temporarily require all probes to upload only Beacon packets, because only they contain synchronization information. If coverage problems need to be identified, only packets with CRC errors followed by their immediate ACKs need to be uploaded.

6.3 Network Topology Reconstruction

As we mentioned in Section 5.1, it is possible to construct a visibility graph of the nodes in the network. One can also attempt to reconstruct the network topology, not just the visibility graph. If we assume that the range of a node is a circle and that the relation between signal strength and distance from the node is known, then we can roughly calculate the position of other nodes with respect to the probes. If a node is visible from three or more probes then we can calculate its position by triangulation.

However, the relation between signal strength and distance is not precise and so, for a given signal strength received at a probe, there will be a disc (instead of a circle) on which the node could lie. Even three probes would not be sufficient to pin-point the location of a node. But we will get a region in which the node possibly lies. In this way, we can build a rough map of the network topology.

7 Conclusion

Distributed trace collection and cross-correlation is a powerful technique that enables us to detect many problem sources in a wireless network. It offers considerably more information that a sniffer running on a single laptop. Having traces from all parts of the network makes it possible to diagnose problems which could not have been noticed from a single sniffer trace. More importantly, we have demonstrated that such a technique can be implemented as an automated tool.

We have implemented the automatic trace collection utility and have analyzed sample traces employing the algorithms discussed above. Our implementation is off-line but we have shown how to carry out similar steps in real time. A network monitoring system working in real time, and possibly enhanced with visual information in terms of a topology map which is also obtainable from this approach, will be of immense use for a network administrator for fine tuning his wireless network.

Acknowledgment

We are thankful to Dr. Pravin Bhagwat for his invaluable suggestions and constant guidance without which this work would not have been possible. We are also grateful to Dr. Dheeraj Sanghi for guiding us during the initial phase of this project. We would also like to thank our colleagues Kamalika and Siddhartha, who worked with us on the implementation of the trace collection utility.

References

[1] IEEE 802.11, *Wireless LAN MAC and PHY Specifications*, http://standards.ieee.org/getieee802/802.11.html

[2] Paxson V, *End-to-End Internet packet dynamics*, ACM SIGCOMM '97

[3] *Netperf: A network performance benchmark*, http://www.netperf.org/