In-Kernel Passive Measurement of the Performance Impact of Hidden Terminals in 802.11 Wireless Networks

Scott Raynel
RuralLink, Waikato Innovation Park
Ruakura Lane, Hamilton
New Zealand
scott@ruralink.co.nz

Tony McGregor
Computer Science Department
The University of Waikato
Hamilton, New Zealand
tonym@cs.waikato.ac.nz

ABSTRACT

The negative performance impact of the presence of hidden terminals in wireless networks has been well known for decades. Despite much research in the area, many deployed networks continue to suffer a performance penalty because of hidden terminals. Ad hoc wireless networks are particularly susceptible to hidden terminal collisions because there are fewer opportunities to plan the network in a way that avoids or reduces the number of hidden terminals. Measuring the presence of hidden terminals and the impact they are having on performance is difficult, especially in a network of many nodes. Without such measurements, the users and operators of wireless networks can not tell if performance problems are caused by hidden terminals or some other problem. We introduce new methodology that can detect the presence of hidden terminals and estimate the performance impact they are causing. The methodology requires no additional hardware and is suitable for wide scale deployment and long term operation. The approach is based on in-kernel instrumentation of the wireless network stack. The design, implementation, and testing of the approach are covered. Results from in-lab testing and the measurement of a live commercial 802.11 network are also presented, including a case study where performance was significantly improved.

Categories and Subject Descriptors
C.2.3 [Computer Systems Organisation]: Computer-Communication Networks/Network Operations[Network management, Network monitoring]

General Terms
Measurement, Performance

Keywords
Hidden Terminal Collisions, 802.11, Wireless networks, Passive measurement.

This is the authors version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in PE-WASUN’11, November 3-4, 2011, Miami, Florida, USA.

1. INTRODUCTION

The “hidden terminal problem” is a classic performance challenge faced by CSMA/CA networks.[23]. Despite much research over decades (e.g. [3, 8, 14, 25, 17, 24, 7, 9, 1]), many deployed networks, including ad hoc networks, suffer performance degradation from hidden terminal collisions. Hidden terminals are but one of many potential causes of poor performance in wireless networks. For users and operators of these networks, a critical issue is determining the particular cause of poor performance. Existing methodologies for wireless performance measurement either deliver too broad based information (e.g. latency, loss and throughput) to identify a particular problem, or are only indirectly related to the actual performance seen (e.g. Signal to Noise ratio) or are expensive in time and/or resources to deploy (e.g. external passive measurement).

We present a new methodology for measuring wireless networks and demonstrate its utility by measuring the existence of hidden terminals and their current performance impact. The methodology is based on instrumenting the wireless network stack in the operating system of the wireless devices. Our implementation is for Linux based 802.11b routers using the MadWiFi stack because these are open source products. However, the methodology could be applied to most wireless devices (except, perhaps, wireless sensor networks which have very limited memory and CPU resources) with the support of the vendor.

2. BACKGROUND

There are other approaches to measuring the effect of hidden terminals. In 2004, Raya et al. presented DOMINO [18], a system for detection of greedy behaviour in IEEE 802.11 wireless networks. It uses a passive external monitor at an Access Point (AP) to detect anomalies in client back off times. They specifically focus on detecting clients with modified 802.11 contention window parameters that allow unfair access to the medium. DOMINO is an entirely passive system and does not require modification of the client devices. While the original intent of DOMINO was focused on detecting greedy client behaviour, the same principle could be used to detect the transmissions of hidden terminals.

The MOJO project [21] diagnoses physical layer anomalies that cause problems in wireless networks. It does so by augmenting client devices with additional hardware and software which performs measurements and collates the data at the AP. The APs can then apply detection algorithms to find hidden terminals and also signal and noise level variations.
Li et al. [11] use a combined passive and active technique for detecting hidden terminals. The technique records CTS frames and keeps track of the number of stations that are not directly connected. This passive method will not always give an accurate count because some terminals may not be transmitting CTS frames. To generate a complete picture of the neighbourhood, occasionally a probe frame is injected to solicit a response from each terminal’s two-hop neighbours. This method is simple enough to be implemented within the MAC and appears to be intended for use by a wireless network stack to build a hidden station table so that the MAC can determine if RTS/CTS is necessary for each frame.

2.1 Measurement Challenges

Passive external capture systems, like DOMINO and MOJO, rely on “vicinity sniffing” [26, 6]. In this technique, the capture device is physically separate from the network node being measured so it does not introduce any additional overhead. However, this separation means that the monitor will not see exactly the same set of events as the terminal being measured. For example, environmental effects of multi-path, different RF front ends and different antenna may cause the monitor to see different channel conditions to the measured node. Schulman, et al. [20] and Yeo, et al. [27] investigated the “fidelity” of traces captured using external passive wireless monitors and found that many traces are incomplete or inaccurate. Passive external capture is also not feasible for most long term and network wide measurement. It requires extra hardware at all capture points.

An alternative approach is to run capture software on the terminal being measured. Wireless nodes are often based on cheap, low power hardware platforms and may be sensitive to additional CPU load and memory use. Tools that might be used on larger machines (e.g. libpcap and tcpdump [22]) are less useful on this type of hardware. For example, they often copy packets from kernel space to user space before they are analysed. Some tools are available to reduce this cost but they are limited in their utility. For example, libpcap and tcpdump provide an interface to the Berkeley Packet Filter (BPF) [13]. It can reduce the overhead of copying all packets into user space by selecting a sub-set of packets that should be analysed. However, BPF does not provide enough flexibility for many monitoring applications. For example, it does not provide a method for random packet sampling and can not summarise data from the packets selected. We propose the Wireless Measurement Project (WMP) framework to address these issues.

3. THE WMP FRAMEWORK

The WMP Kernel Measurement Framework performs measurement in a Linux kernel module by instrumenting the MadWiFi wireless device driver. A similar technique is used in [12] and [21] (MOJO). MOJO is the closet to a approach to WMP and includes hidden terminal detection. However, it uses additional radio hardware on each client device which limits its scope and accuracy two different radios, even very closely located, will not always see the same frames. The WMP architecture (see figure 1) splits the measurement task between the WMP-core, which instruments the MadWiFi driver, a WMP kernel module and a user space helper. By using kernel modules and de-coupling the measurement modules from the wireless device driver, WMP modules that implement particular measurement tasks can be dynamically loaded and unloaded at run time without affecting the forwarding of packets.

The current implementation uses the MadWiFi driver although we do not expect problems using other open wireless stacks like, for example, Linux’s generic mac80211 stack. In addition to the hidden terminal measurement work described here, WMP been used to provide network wide long term link level statistics for a study of the accuracy of traditional path loss prediction models. [15].

3.1 Driver Hooks

At the lowest level of the framework, modifications are made to a wireless NIC device driver to enable extraction of relevant information about each MPDU as it passes through the driver. Once the data has been collected, the driver passes the frame and its meta-data to the appropriate WMP kernel modules. Measurement modules are registered with the WMP core when they are loaded into the kernel. Registration includes the physical device that the module wishes to monitor.

In normal operation, when an incoming frame is received on the RF side of the NIC it is decoded and placed in a hardware receive queue. The NIC generates interrupts which are handled by the wireless device driver running on the host. During interrupt processing, frames are transferred from the device to host memory. Multiple frames may be transferred to the host during a single interrupt to reduce the overhead related to handling an interrupt. The NIC provides a hardware time stamp with each frame. Once the driver has a frame from the physical device, it is dispatched to the appropriate logical network device.

At the same time as a frame is dispatched to a logical network device, the WMP core also passes a reference to the frame to the measurement modules that have registered that they want frames from that physical device. In the transmit path, frames and meta-data are captured by WMP once the frame has been transmitted when a “transmission complete” interrupt received.

3.2 Measurement Module API

Measurement modules interact with the WMP-core through a simple API. The API is passed on registration and a callback function which is called by the WMP core each time a frame is received or transmitted on the selected device. The
API presents a consistent set of meta-data about each frame processed by the driver along with a zero-copy reference to the frame content.

The frame data provided to a measurement module includes the IEEE 802.11 MAC header as seen on the medium. The header includes potentially interesting information such as the retry bit and sequence number which can indicate the presence of link-level retransmissions. The meta-data block passed to the module contains information about the frame including the hardware time stamp, bit rate, antenna, and other information about the frame’s reception.

Multiple modules may register with the WMP-core for the same device. In this case more than one module will receive a reference to a frame. When a module is finished with a frame, it returns from its call back. The WMP-core manages release of the frame data when there are no more users.

In addition to performing its own measurement tasks, a module can indicate whether the packet should be passed to a “monitor mode” logical network interface, if one exists. Monitor mode interfaces pass a raw copy of frames to user space via a Radiotap [16] or Prism II header. They usually include packet meta-data. Monitor mode interfaces are used in traditional passive external and internal capture and it is the copying of each frame into user space that introduces a significant overhead and frame loss on resource constrained systems. For some measurements, a copy in user space of some frames may be useful. Unlike BPF, a WMP module may contain arbitrary code and allows frame selection algorithms to be implemented more simply but support more sophisticated selection. For an example of the challenges of selecting a random set of frames with BPF, see [19].

### 3.3 Data Collection

User space programs extract data from a WMP using the standard Linux kernel interfaces. These include: `/proc` [4], `/dev` and NETLINK socket [5].

### 3.4 Comparison to Passive External Capture

When compared with passive external capture, WMP improves accuracy in three ways. First, the frames that are seen by the measurement system are the exact same frames that were decoded by the measured host. Secondly, avoiding copying all frames to user space reduces frame loss and consequential uncertainty in the measurement results. For more detail of the impact of this problem, see Schulman et al. [20] who found that many passive captures were incomplete due to overloading on the sniffer causing frames to be dropped. Finally, no extra hardware is required so it becomes feasible to measure on a wide scale (possibly even at every interface in a network) and for long durations (possibly even permanently, depending on the amount of data generated).

## 4. Measuring Hidden Terminal Collisions

Two methods for the detection of hidden terminals have been implemented using WMP. The first uses network connectivity discovery to identify the potential for hidden terminal collisions. The second uses violations of the IEEE 802.11 Distributed Coordination Function (DCF) to detect instances of hidden terminal collisions in an operational network and, from them, estimate the performance impact of the hidden terminal collisions that are occurring.

The two methods complement one another. The first indicates that hidden terminal performance degradation is possible but not to what extent it is currently impacting the network. It is possible that, because of the traffic profile of the network, potential problems do not have any significant performance impact at the moment. The second method measures the current performance impact of hidden terminal collisions but can not indicate whether there are areas of the network that currently experience little or no impact but may, at another time.

### 4.1 Network Connectivity

The hidden terminal problem occurs when two or more terminals are unable to detect each others carrier but are heard by a common third terminal. A directed graph can be used to model the connectedness of a network. Each vertex in the connectivity graph indicates a terminal and each directed edge indicates the ability of the terminal at the head of the edge to carrier sense the terminal at the tail. A strongly (fully) connected graph indicates no hidden terminals. Hidden terminals in a weakly connected graph can be found by examining the edges between sets of vertices. The absence of an edge between two vertices with outgoing edges to a common third vertex indicates a hidden terminal.

Measuring the connectedness of a network based on carrier sensing using only the commodity wireless hardware is not always possible. Sensing just a carrier is enough to avoid

<table>
<thead>
<tr>
<th>Call</th>
<th>Direction</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>wmp_register_monitor_handler</td>
<td>to core</td>
<td>Register module to receive frames for a device. Includes the call back function and an opaque pointer.</td>
</tr>
<tr>
<td>wmp_unregister_monitor_handler</td>
<td>to core</td>
<td>Unregister a modules from a device</td>
</tr>
<tr>
<td>Module call back</td>
<td>from core</td>
<td>Passes frame to registered module. Parameters are: the opaque pointer from the registration, a pointer to the Linux device structure (struct net_device), a pointer to the frame (struct sk_buff), a direction flag and a pointer to the frame meta-data block. The call returns either WMP_FRAME_DROP or WMP_FRAME_SAMPLE to control passing the frame to user space via the Radiotap device.</td>
</tr>
</tbody>
</table>

Table 1: WMP-core API
a collision but a frame must be decoded to determine its source. Without resolving this issue, the technique is still useful because it generates a worst case potential for hidden terminal collisions. In future work, this issue may be reduced by sending beacon frames with known characteristics, including frame length and inter-frame spacing. Detecting just the carrier of these beacons is enough to identify the sending station if the frame characteristics are unique over the set of stations that it is possible for a receiver to detect. This is more challenging that it appears at first sight because commodity hardware does not report frames it cannot decode. However, these frames do cause collision avoidance and therefore they can be detected in software.

4.2 DCF Violations

The shortest space between frames in an IEEE 802.11 network where every terminal can sense the carrier of every other is the Short Inter Frame Space (SIFS). The SIFS is the length of time, without any carrier detectable, that a terminal that has a control frame (an ACK for example) to send, must wait before sending the frame. When there are hidden terminals, some of the collisions that occur will violate this requirement. The total number of hidden terminal collisions can be estimated from the number of SIFS violating collisions if the relationship between the length of the SIFS and the total collision space is known.

Collisions may occur during the preamble, the Physical Layer Convergence Protocol (PLCP) header or frame data (PSDU). In these cases the outcome depends on the characteristics of the two received signals. The capture effect (PSDU). In these cases the outcome depends on the characteristics of the two received signals. The capture effect may cause one frame to be received at the expense of the other or both may be lost. However, when a SIFS violation occurs, both frames from the hidden terminals will be received by the common terminal. A WMP module that receives the frames and the timestamps can measure the time at which the PSDU can vary for two reasons: the number of octets in the MPDU payload may vary and the transmission bit rate used to encode the PSDU may vary because of rate adaptation. As a consequence, the length of the collision window varies, however the length of the violation window remains constant. Taking this into account the probability of a DCF violation for variable length frames can be estimated by substituting \( p' \) into Equation 4.

\[
p' = \frac{\sum i t_i + v}{\pi n}
\]

where \( t_i \) is the length of frame \( i \). The total rate of collisions due to hidden terminals with variable length frames can be estimated by substituting \( p' \) into Equation 4.

5. IMPLEMENTATION

5.1 Network Connectivity

The connectivity graph is built from data collected by a WMP kernel measurement module on each host in the network that puts the NIC into promiscuous mode. The module builds a list of sender addresses it has seen. While the methodology can run entirely passively, this introduces the risk of not observing a terminal that is currently inactive or that have a very high error rate. To address this, an active component sends occasional beacon frames. Beacons

Figure 2: The relationships between a PPDU, its ACK window and its violation window. Any PPDU beginning within the violation window is from a terminal that is unsynchronised with the transmitter of the original PPDU.
contain a forward error corrected payload which normally allows frames to be identified even if they are received in error. The module exports the list of terminals seen via a /proc file. User processes collect and analyse this data from across the network.

5.2 DCF Violations

Detecting DCF violations requires measurement of the inter-frame gap. The NIC driver only reports one time stamp for received frames: the time when the last byte of the frame was received. Figure 3 shows an IEEE 802.11b HR/DSSS/long PPDU. The length (in time) of a frame is given by the \( \text{TXTIME} \) functions in the 802.11 specifications.

\[
\text{TXTIME} = \text{PreambleLength} + \text{PLCPHeaderLength} + \left\lceil \frac{\text{LENGTH} \times 8}{\text{DATARATE}} \right\rceil
\]

In this case

\[
\text{TXTIME} = \text{PreambleLength} + \text{PLCPHeaderLength} + \left\lceil \frac{\text{LENGTH} \times 8}{\text{DATARATE}} \right\rceil
\]

where

- **PreambleLength** and **PLCPHeaderLength** are dependent on whether a long or short preamble is used and are 72µs and 24µs for a short preamble or 144µs and 48µs for a long preamble.
- **LENGTH** is the MPDU length in octets and **DATARATE** is the PSDU encoding data rate in units of Mbit/s.

The inter-frame space is calculated from the end time of two consecutive frames and the length in time of the second frame. Several heuristics are used to improve the DCF violation detection accuracy. Frames are checked for truncation, to avoid miss-classification of collisions as DCF violations, and to determine if they have a long or a short preamble.

5.2.1 Truncated frames

A truncated frame could be easily detected if the expected length of the frame was available. While the PLCP header includes a length field that contains the MSDU field length in microseconds, this information is generally not made available to the host operating system by the NIC hardware.

Truncation detection operates as follows: frames received without an FCS error can not be truncated. If the frame is less than the minimum length of an IEEE 802.11 MAC frame, it must be truncated. In other frames, common MPDU payloads, including LLC/SNAP, ARP, IP, PPPoE, and EAPOL, include length information. This is used to detect other cases of truncated frames.

5.2.2 Preamble length

PPDUs sent with a PSDU encoding rate of 1Mbit/s are sent with a long PLCP preamble but others may have a long or short preamble. The method begins by assuming that a long preamble was used and the inter-frame spacing is calculated. An inter-frame spacing with a negative magnitude indicates a short preamble. Terminals do not change the preamble settings in normal operation so, if a short preamble is detected, this is assumed for other frames with the same characteristics that are sent by that terminal.

The module records every packet pair with a DCF violation. These are then exported in CSV format via /proc. The file starts with a header containing meta-data. This includes the number of packets seen, the number of events detected, the number of records dropped due to memory constraints, the device’s hardware address, its interface name, and the total airtime occupied by the frames received.

6. DEPLOYMENT EXPERIENCE

Both methodologies have been deployed in a commercial wireless network (operated by RuralLink) that provides service to rural customers. The DCF violation methodology was also tested in the laboratory.

6.1 Laboratory Testing

The primary purpose of the laboratory testing was to validate that the relationship between the number of DCF violations and the total number of hidden terminal collisions predicted by equations 4 and 5 in a controlled setting. Laboratory tests also characterised the accuracy and precision of the NIC’s hardware timestamps.

The equipment used was: Atheros 5212 based mini-PCI 802.11abg wireless NICs, Soekris 4526 hosts for the hidden transmitting terminals and a Soekris 5501 microcomputer for the common receiving terminal. The 5501 did not drop any frames due to buffer overruns. The Soekris microcomputers ran a customised Linux distribution based on Debian Sarge, the 2.6.16 kernel and the MadWiFi Atheros Linux driver version 0.9.4. The transmitting terminals were “hidden” from one another by disabling the CCA function of the Atheros NICs (see [2]).

A series of experiments were run with different inter-frame spacing and frame sizes. The results are shown in figure 4 which indicates a strong match between theory and practice in the laboratory.
Figure 5: Connectivity graph for PKU site. Note that several other site interfaces can be heard by the PKU interface under measurement, including another interface on the same host. Directly connected clients are indicated in yellow. Non-directly connected clients are interfaces are indicated in black. Most of the client CPEs are poorly connected to each other, and there are several client CPEs visible that are not directly connected to the PKU site, i.e. they are connected to another site but their transmissions can be decoded by the PKU site interface.

Figure 6: Indicative traffic profile of the PKU access point interface over a 24hr period with 5 minute averages.

6.2 Live Network Testing

The RuralLink network is primarily made up of 5.8GHz point-to-point backbone links and a 2.4GHz point-to-multipoint access network, based on IEEE 802.11 technology. Client CPEs and the backbone routers run a customised GNU/Linux operating system and use Atheros 802.11 chip sets. Results for over 100 interfaces were collected. A range of interfaces, with different degrees of connectivity, were studied in detail. The interfaces studied were not selected in any other way. The low connectivity example (the “PKU” repeater site in Te Awamutu, NZ) is presented here. A typical traffic profile for the PKU interface is shown in fig 6.

The network connectivity method (see fig 5) showed 22 transmitters. The 11 directly connected clients are poorly connected to one another because they are separated by long distances. The other 11 transmitters were not part of the network.

The timing method indicated a high rate of hidden terminal collisions, in the order of 5-14% of frames (see fig 7). Further investigation showed that the non-directly connected terminals were transmitting on the same channel as the terminal in question and were contributing significantly to the number of detected hidden terminal collisions. With this in mind, the channel allocation scheme was modified to reduce channel overlap. Figure 8 shows the connectivity after the change. The number of collisions decreased significantly, as shown in Figure 9, and lead to a significant performance improvement.

In this case, the combination of the connectivity and DCF violation methods provided information that enabled network performance to be significantly improved. While this example is based on a network with low connectivity, high and medium connectivity examples also exhibited the expected characteristics, as did a selection of other interfaces.

7. CONCLUSION

Hidden terminal collisions are an important cause of performance degradation in wireless networks. There are strategies that can reduce the performance impact of caused by hidden terminal collisions. However, there are many potential causes of poor performance in wireless networks. Isolating the impact of hidden terminal collisions in a particular network is a pre-requisite step for addressing these problems.
Figure 8: Connectivity graph at PKU after channel change. Only two non-directly connected terminals are now present in the graph. The connectivity between directly connected client CPEs remains similar, as expected. Hidden terminal collisions are still possible due to poor connectivity between directly connected terminals.

WMP is an in-kernel, passive measurement framework that allows wide spread, low cost and flexible measurement of wireless networks.

Using WMP, we have developed two approaches to measurement of hidden terminal collisions in a wireless network. The first measures the potential for hidden terminal collisions and the second the performance impact under current conditions. Their utility was demonstrated through a combination of in-lab testing and deployment in a live wireless network providing service to users. In our initial deployment, the methodology highlighted a performance problem that was resolved through a simple network reconfiguration. This resulted in a large improvement in performance.

8. ACKNOWLEDGEMENTS

The authors would like to thank RuralLink for access to their network and support to deploy the measurement software. This was invaluable for this work. We gratefully acknowledge The University of Waikato support for this work through a PhD scholarship to the primary author.

9. REFERENCES


