Rate Adaptation for 802.11 Wireless Networks: Minstrel

Paper #86, 14 pages.

ABSTRACT

Minstrel is a practical rate selection algorithm for commodity 802.11 radios, implemented in the Linux kernel and the ns-3 network simulator. Design of a rate selection mechanism is complicated by the variety of sources of packet loss in 802.11 networks, the difficulty of testing rate adaptation, and interaction with higher layer protocols (especially TCP). Minstrel is based solely on acknowledgement feedback. Consequently, estimates of future success at a given rate are based only on past success ratios at that rate. Minstrel selects rates that will give an approximation to the best available throughput, while delivering packets with the highest probability achievable given link conditions and a constraint on the transmitter time to be spent on any one packet, so as to be friendly to higher layer protocols. If Minstrel fails to achieve communication with a neighbour, it collects good evidence that no communication is possible. Conversely, if communication is at all possible, Minstrel will find rates to use that will achieve the best reliability and close to the best performance available.

Categories and Subject Descriptors

C.2.1 [Network Architecture and Design]: Wireless communication—Rate adaptation; C.2.2 [Network Protocols]: 802.11—Rate adaptation

General Terms

Algorithms, Performance

Keywords

802.11, Rate Adaptation

1. INTRODUCTION

In early 2006, we were assessing radio cards to upgrade a large outdoor wireless mesh network from 802.11b to 802.11g, when we realised that the available rate adaptation algorithms for 802.11g cards in Linux were giving rather poor performance and even worse reliability. One reason for this was that the outdoor links had high levels of packet loss and, like TCP, the rate adaptation algorithms were interpreting packet loss to mean they should

slow down, whereas manually fixing high rates showed conclusively that those high rates did in fact work well. Over the next year, we developed and thoroughly tested a replacement rate adaptation algorithm. Our algorithm, Minstrel, achieves high performance and the highest possible robustness while remaining interoperable with 802.11. All other available rate control algorithms have severe robustness problems, causing links to be unusable. Minstrel became the default solution in the Linux kernel during 2009 due to its high performance and general robustness. In those three years we learned a great deal about rate adaptation, interactions between 802.11 and higher layer protocols, sources of packet loss in 802.11 systems, and how to test adaptation algorithms.

This paper will give a general introduction to 802.11 rate adaptation and sources of frame loss in 802.11 networks. We then discuss design considerations for Minstrel. We make some initial comparisons with a selection of rate adaptation algorithms, covering the three principal families of algorithms (step up/step down, SINR, and acknowledgement feedback estimation). Minstrel is described in detail. We explain our testing procedures, then present some simulation results to give a detailed view of some of the issues with rate adaptation. Finally we summarise the failure modes of previous algorithms.

2. 802.11 RATE ADAPTATION

The IEEE 802.11 wireless networking standard specifies PHY devices that support multiple rates for transmission of physical layer frames, but it does not define an algorithm for selection of transmission rates, nor does it fully define the retransmission mechanism:

Some PHYs have multiple data transfer rate capabilities that allow implementations to perform dynamic rate switching with the objective of improving performance. The algorithm for performing rate switching is beyond the scope of this standard, but in order to ensure coexistence and interoperability on multirate-capable PHYs, this standard defines a set of rules to be followed by all STAs. [IEEE Std 802.11 2007]

The rate selection problem is to select a rate for each transmitted unicast data packet that gives the best performance. Brodsky and Morris [2009] show that rate adaptation is a critical part of performance for 802.11-like radio systems, and yet many rate selection algorithms can be out-performed on some links by a well-chosen manually selected rate, at least for a time. It is possible to characterise the rate selection algorithm as automating the process of an operator watching the performance of a link and manually adjusting a fixed rate to achieve best performance. Automatic rate control should never be significantly worse than the performance of the best manually selected rate for a certain link.

There are five principal sources of packet loss due to channel characteristics at a given rate:

- Interference due to noise, other networks, or system self-interference due to distant parts of an IBSS network.
- Front end saturation or automatic gain control desensitisation due to high power transmission near the receiver (possibly on other channels or out of band).
- Inter-symbol interference due to multipath propagation.
- Insufficient received energy due to path loss or fading.
- Local losses in the receiving radio device or receiving system.

The first two can be modeled as a rate of packet loss, not necessarily uniform, due to packets being 'shot down'. The third and fourth can be modelled as a link with both slow and fast fading. Due to multipath from moving objects in the environment, even if the network nodes themselves do not move, 802.11 channels are frequently fast-fading channels with interference-induced random packet loss.

Salyers, Striegel, and Poellabauer [2008] show that there are sources of loss local to the radio receivers, and that these losses can amount to 50% of the losses on real links. Their observation may be partially explained by differing responses to errors in the low bit rate PLCP header that precedes an 802.11 frame to signal the data rate and transmission time for the remainder of the frame [IEEE Std 802.11 2007, Fig. 15.6]. Device driver bugs and memory pressure in the receiving node also fall into the category of local loss. A rate control algorithm must be stable in the face of potentially severe local loss or PLCP header errors.

One local packet loss issue for ad-hoc (IBSS) networks is beacon desynchronisation due to propagation delay and delays in beacon processing within the MAC and driver. Since the transmit queue must be stopped during the beacon interval, if the beacons are desynchronised the restart may be delayed, inducing long round trip times. On a busy link, buffers are likely to fill up and cause packet loss. There is a long-standing beacon desynchronisation bug in ad-hoc and mesh drivers for Atheros hardware on most platforms that can lead to bursts of 90% packet loss and single-hop round trip times of up to a beacon interval, a phenomenon called 'ramping' [Lukas 2007]. 'Ramping' is named after the sawtooth pattern visible in plots of link latency vs time, and has nothing to do with the mechanism of the bug. This bug is likely to confound the interpretation of many experimental performance results in this field that use Atheros hardware in ad-hoc mode; it was certainly difficult to interpret our own tests during Minstrel development. The drivers used to present this research do not exhibit this bug, although it is important to note that we only found a solution to the ramping bug after we had Minstrel to render the links robust enough to gather the necessary data¹. In a long-range outdoor network, some (relatively minor) beacon desynchronisation is inevitable given the design of the 802.11 MAC, although this does not have serious performance consequences².

Even indoors at short ranges, rate adaptation can be necessary for good performance. For example, a user may move a portable device a short distance, which happens to place an obstacle between the device and an associated access point. Rate adaptation should quickly respond to that situation, dropping no more than a few packets in the process. Similarly, moving the device out of the shadow should quickly result in an increase in performance. In either case the user's TCP connections should not go into long retransmit timeouts, since this is typically interpreted by users as 'the network has failed'.

In an indoor environment, severe high-speed multipath fading may still occur; consider a coffee shop hotspot, with large windows facing the street. Vehicle traffic will create intense and potentially high-speed multipath fading even when the access point and client are stationary. This can occur even if the moving vehicles are well outside the practical coverage of the access point.

It is interesting to examine how the main sources of packet loss vary according to data rate. Interference sensitivity increases with the rate while saturation is independent of rate. Presuming that a large proportion of interference consists of uniform periodic transmissions, losses in both cases are proportional to transmission time. By our observation most of the interference present on

¹The 'ramping' bug is present in all open-source drivers for any Atheros hardware at the time of writing, in some drivers for Microsoft Windows, and potentially in drivers for other systems as well. Patches for the Linux mac80211 drivers ath5k and ath9k are in preparation.

²Beacon desynchronisation is also possible in 802.11s mesh networks, however we have not investigated the consequences in this case.

outdoor urban links consists of periodic transmissions from 802.11 access point beacons and continuous transmissions from analog video cameras. This leads to higher losses at lower rates due to these mechanisms. Intersymbol interference is uncorrelated between rates. Link propagation loss, whether due to fading or range, affects higher rates more than lower in general, although not necessarily to a very large extent since there may be no change in receiver sensitivity across the majority of the 802.11bg rates, and only 3 dB variation across all rates.

3. DESIGN CONSIDERATIONS FOR MIN-STREL

Minstrel is part of a larger multi-year project, in which we designed and built a number of wireless mesh networks using embedded Linux routers with 802.11 radio cards in IBSS mode, routing with OSPF MANET [Henderson et al. 2003]. These networks are used for internet and intranet traffic, and intended to reach commercially acceptable reliability and performance levels. This implies that the network nodes are required to be stable over periods of months with no downtime except for scheduled maintenance and power outages. It is regarded as unacceptable in this environment to work around instability in wireless drivers or rate algorithms by bringing interfaces down or rebooting nodes to return performance to nominal.

The standard network node in these networks uses an Atheros 802.11g radio card³ with a 1 W amplifier and 8 dBi colinear dipole antenna on a platform using an IDT RC32434 MIPS embedded CPU and running an in-house Linux distribution, including a version of the Madwifi⁴ driver with all support for non-IBSS features removed [Leffler and Renzmann 2008]. Individual links range up to approximately 2.5 km, and the networks are usually in urban areas, and therefore subject to a variety of interference sources, most notably other 802.11 networks. It is necessary to adjust various timing parameters of the 802.11 stack, especially the slot time, away from standards-compliant values in order to achieve correct operation at ranges beyond 300 m. Since parts of the MAC are implemented in hardware, this constrains choice of radio cards for these applications.

Development of Minstrel was prompted by the observation that the available rate control algorithms for the Madwifi driver for Atheros 802.11g radio cards [Leffler and Renzmann 2008] were not delivering performance comparable to short-term manual rate selection in many cases, and the remaining cases appeared to be due to the link changing too fast for a human operator to keep up. Section 4 details our conclusions on rate control

algorithms we had available to us. The tests described in Section 6 had been adopted within our project long before we commenced work on Minstrel, and those tests largely drove the development of Minstrel. Development of Minstrel commenced in March 2006, and the algorithm had close to its present form by March 2007.

Minstrel was primarily intended to address rate control for outdoor environments, where RSSI and SINR are poor guides to rate success [Aguayo et al. 2004, Brodsky and Morris 2009, Vutukuru et al. 2009] and therefore the rate adaptation problem is particularly difficult. The key problem is: what information is available to guide rate control and how reliable is that information? Aside from signal strength and channel noise measurement, the only other feedback available in commodity hardware is reception of acknowledgement frames from the neighbour. This is the only source of channel and remote neighbour information used by Minstrel.

Certain parts of the 802.11 protocol are implemented in hardware, especially including the acknowledgement and link-layer retry mechanisms⁵. 802.11 radios often also support multi-rate retry, in which retries may be attempted automatically in hardware at multiple rates on a schedule supplied by the driver software. Therefore, rate selection is required for retry rates. Acknowledgement feedback is returned giving the number of transmission attempts for each frame, and combining this with the multi-rate retry schedule allows unambiguous knowledge of which attempt at which rate succeeded.

The design of Minstrel was guided by several observations:

- A rate adaptation algorithm should never render a link unusable if there is any possibility of communication.
- All practical protocols that can be used over 802.11
 have some mechanism for mitigating packet loss,
 but all such mechanisms have a performance cost.
 Therefore delivery probability must be maximised,
 subject to the other constraints.
- TCP behaviour over wireless links can be problematic, especially in the presence of even moderate packet loss rates. In particular, TCP stalls due to sequential lost packets lead to long periods of no link utilisation.
- Many protocols, including TCP, are sensitive to round-trip-time, and especially to RTT variation.
 RTT 'spikes' are caused by head of line blocking if too many retransmissions are requested at slow data rates. Therefore it is appropriate to bound the time spent working on any one packet rather

³Senao NMP-8602 ETSI, Atheros AR5413 chipset

⁴Madwifi is a Linux driver for Atheros 802.11g radio cards, now obsoleted for general use by the mac80211 based ath5k driver.

⁵Retransmission in the 802.11 MAC happens on a very short timescale, too short for implementation in current general purpose operating systems.

than using a constant number of retries. Bounding the time working on each packet will increase the probability of packet loss on a poor link, but even then may improve application performance.

- Any single rate will very likely show bursty transmission success (acknowledgement) probability, but
 those bursts occur independently on each rate, and
 therefore the maximum throughput possible with
 any given static link is fairly stable over time.
- A rate control algorithm for hardware supporting multi-rate retry should have some reasonable basis for selecting retry rates and number of retries, consistent with the diverse loss sources. Maximising delivery probability within a time bound is one such basis that also accounts for the characteristics of higher layers in the protocol stack.

Minstrel attempts to select rates that will give an approximation to the best available throughput, while delivering packets with the highest probability achievable given link conditions and a constraint on the transmitter time to be spent on any one packet. Minstrel assumes that rates for those frames that are constrained by the 802.11 specification will be assigned by other parts of the 802.11 stack, and therefore only solves the part of the rate selection problem not covered by the specification.

4. OTHER RATE ALGORITHMS

Receive sensitivities monotonically decrease with increasing data rate for many 802.11 radio chipsets, which suggests a simple rate algorithm. Essentially, estimate packet loss rate, and if the packet loss rate drops below a threshold, step up a rate, if above a slightly lower threshold, step down a rate.

This simple step up/step down algorithm in practice produces poor performance, especially in the presence of fading or interference [Wong et al. 2006]. Clearly, if the dominant loss mechanism causes loss in proportion to the transmission time, stepping down on excessive packet loss will only make the situation worse, which leads to selecting the slowest rate and, paradoxically, the highest loss rate. Once, while now obsolete, was a common example of a step up/step down algorithm [Wong et al. 2006]. PID, which was the previous default in Linux, is also an adaptive step up/step down algorithm [Brivio 2008]. Wong, Lu, Yang, and Bharghavan [2006] proposed RRAA, which is another algorithm in this class.

Similarly, rate control algorithms based on received signal strength and noise level measurement and knowledge of the radio parameters also perform poorly in the presence of noise and/or fast fading [Rodriguez 2009a, Rodriguez 2009b, Wong et al. 2006, Vutukuru et al. 2009]. In such an environment, noise measurement suffers from sampling issues, as noise measurements are typically

taken infrequently compared to packet transmissions elsewhere in the vicinity. The wireless development office environment described by Rodriguez [2009a], with large numbers of access points in the vicinity, roughly corresponds to a large conference network with many hidden nodes. Large numbers of access points are frequently receivable from outdoor antennas in urban areas; we have received beacons from more than 300 different access points with a single stationary 8 dBi omnidirectional antenna. 'Ideal' is an algorithm implemented in the ns-3 network simulator that serves as an example of an SNR or SINR based algorithm [Henderson et al. 2009].

Received signal strength measurement has a further issue in that paths are not necessarily symmetric. Regardless of channel reciprocity, each station's noise environment is unique, and if the remote node has a different antenna or different output power, simply using received signal strength from that node will likely result in the selection of rates that are less than optimal. Brodsky and Morris [2009] show that the presence of multipath implies that there can be as much as 14 dB uncertainty in a sender's ability to estimate SNR at the receiver, even given good information on the power output and antenna characteristics. Wong, Lu, Yang, and Bharghavan [2006] observe this variability experimentally. In the absence of a protocol feature to return remote signal strength measurements or bit error statistics, signal strength has little value in guiding rate selection.

Many times even in the absence of any mobility, there is no constant rate that performs well for any length of time, as the environment is likely to change. The key issue is that in many environments, practical transmission success is uncorrelated with received signal strength, and all bit rates are uncorrelated with each other. Fortunately, however, transmission success probability at any given bit rate seems to vary slowly enough with time; if it did not, there would be no viable solution to the rate control problem [Aguayo et al. 2004]. Wong, Lu, Yang, and Bharghavan [2006] measured the mutual information of two packets as a function of the time between them, and showed that this decayed over approximately 150 ms. This is a long enough period of time for driver software to respond, since typical hardware delivers new transmission success reports and other information delayed by a few milliseconds.

The SampleRate algorithm attempts to resolve these issues by using acknowledgement feedback [Bicket 2005]. SampleRate tracks acknowledgement success probabilities using an exponential weighted moving average, and selects rates based on those estimated probabilities together with a number of heuristics, targeting minimum transmission time. The callback used to track acknowledgement feedback was added to Madwifi to support SampleRate.

SampleRate works reasonably well for rather static

Per-neighbour structure							
maxtp	Rate index		Maximum throughput rate				
tp2	Rate index		Second highest throughput rate				
maxp	Rate index		Maximum probability rate				
probe	Rate index		Probe rate				
lowest	Rate index		Lowest available rate a				
n_rates	Integer		Number of rates available				
stats_update	Time		Time of last statistics update				
minstrel_rates			List of per-rate structures				
			Per-rate structure				
To de transfer	T						
bitrate	Integer Duration		Data rate in bits per second				
perfect_tx_time	Duration Duration		Time to transmit 1400 byte frame, μ s				
ack_time			Time to acknowledge a frame, μ s				
sample_limit	Integer		Weight for probing frequency				
retry_count	Integer		Maximum retransmits				
retry_count_cts Integer			Maximum retransmits with CTS enabled				
retry_count_rtscts	Integer		Maximum retransmits with RTS/CTS enabled				
adjusted_retry_count Integer			Maximum retransmits when used for probing				
success	Integer		Number of successful frame transmissions in this update interval				
attempts	Integer		Number of attempted frame transmissions in this update interval				
last_success	Integer		Number of successful frame transmissions in previous update interval				
last_attempts	Integer		Number of attempted frame transmissions in previous update interval				
cur_prob	Probability ^{b}		Ratio last_success:last_attempts				
probability	Probability		Current probability estimate (Equation 3)				
cur_tp	Integer		Throughput estimate in bits per second				
succ_hist	Integer		Total number of successful frame transmissions ^c				
att_hist	Integer		Total number of attempted frame transmissions				
Per-interface structure							
ewma_level Integer 75% EWMA, w in Equation 3							
segment_size	Duration	6 ms	Maximum time for a retry chain segment				
update_interval	Duration	100 ms	Interval between statistics updates, Δt in Equation 4				
lookaround rate	Integer	10%	Rate of probing, r in Equation 2				
TOOKALOUNG_Late	megei	10/0	reace of probing, T in Equation 2				

Table 1: Minstrel data structures and default values.

environments, but it tends to become unresponsive to environmental changes after a few hours. Also, SampleRate does not perform well in mobile or fast-fading scenarios [Wong et al. 2006] because it responds very slowly to changes in the channel; training can take up to approximately 30 seconds, no matter what volume of traffic is carried. Since SampleRate is superficially quite similar to Minstrel, in particular the use of the exponential weighted moving average, we compare the two on several points in the algorithm description below.

Examining the implementations of rate control algorithms, there are frequently many heuristic assumptions about the characteristics of either the radios or the channel. While these were found to be useful for particular circumstances and hardware, there is no guarantee they will be useful in general. For example, Wong, Lu, Yang, and Bharghavan [2006] show that one such heuristic in SampleRate caused a significant performance loss in their experiments.

5. THE MINSTREL ALGORITHM

Minstrel was intended to address responsiveness and reliability issues with all other rate algorithms available in open-source drivers at the time. Minstrel is based solely on acknowledgement feedback. Consequently, estimates of future success probabilities⁶ at a given rate are based only on past success ratios at that rate. Each decision made in the algorithm has a clear basis, rather than being based on an arbitrary or heuristic choice.

There are three versions of Minstrel: our original implementation in the Madwifi driver for Atheros 802.11g radio cards [Leffler and Renzmann 2008], the generic version ported by Fietkau [2008] to the mac80211 stack in recent versions of Linux [Smithies 2008] and used by default [Rodriguez 2008, Torvalds 2009], and in the ns-3 network simulator by Nguyen [2009] [Henderson et al. 2006, Henderson et al. 2009]. The algorithm description follows the Linux mac80211 version⁷. For clarity, we do

^alowest is left implicit in the implementations

^bProbabilities are represented as integers in the interval [0,18000]

^cTotals are for reporting only

⁶'Probability estimates' here, but in the code and documentation frequently this is abbreviated to just 'probabilities'.

⁷As included in Linux v2.6.32

not describe the additions for radios that do not support multi-rate retry.

In outline, the algorithm is as follows: A table of acknowledgement probability estimates is maintained per neighbour per rate (Table 1). The ratio of transmission attempts to acknowledgements received is maintained using an exponential weighted moving average to smooth the probability estimation (Section 5.3). On a frequent basis, the table is scanned to find an approximation to the best performing rate and retry chain, and that is used for transmission for the next interval (Sections 5.1, 5.3 and 5.4). With a moderate frequency, frames are selected to probe presently unused rates, and feedback from those probe frames maintains the probability estimates for unused rates so that, should the current best rate deteriorate, there is some basis for immediately selecting a more successful rate (Section 5.2)⁸.

As frames are queued for transmit, Minstrel first updates its statistics tables for the destination if necessary (Sections 5.1 and 5.3), then determines if the queued frame is to be a probe frame (Section 5.2), constructs a multi-rate retry chain (Section 5.4), and returns to the mac80211 stack to complete transmission.

Table 1 lists all the per-rate, per-neighbour and global data structures. Most state is maintained per neighbour, except for some precalculated values that depend only on properties and configuration of the radio interface.

5.1 Throughput approximation

Minstrel uses the success probability estimate to estimate potential throughput. The equation used is

$$T_R = \frac{p_R}{d_R} \tag{1}$$

where T_R is the throughput estimate for rate R, p_R is the success probability estimate for rate R, and d_R is the duration of a single frame transmit attempt at rate R, presuming a 1400 byte frame. Note that this is substantially different from the often-cited TCP rate equation from Padhye, Firoiu, Towsley, and Kurose [1998], which is roughly proportional to $1/\sqrt{1-p_R}$ in this notation. Equation 1 is an estimate of the bandwidth achieved at the link layer, rather than the TCP throughput that should be possible across the link. In the implementations, probability estimates are scaled to the interval [0,18000] and T_R is in units of bits per second for sufficient resolution while avoiding overflow in unsigned 32 bit integers.

This equation somewhat overestimates the actual throughput. A better approximation would add the inter-frame space to d_R . However, all this will achieve is reported rates that better match the rate reported by link mea-

surement tools such as *iperf*, since the actual rates chosen depend only on the order of rates when sorted by T_R .

By comparison with the transmission time equation used by SampleRate, Equation 1 is a much smoother function, allowing Minstrel to resolve much finer distinctions between bit rates and their loss statistics. That SampleRate uses transmission time rather than throughput is unimportant, the important factor is the degree of quantisation involved in the limited precision calculations.

5.2 Probing

The probing mechanism attempts to probe on 10% of frames. 10% was chosen empirically as a tradeoff between intense probing, giving more information to rate selection, and the overhead of probing at rates that do not work.

Frames to use for probing are chosen when the number of probes issued falls behind the configured rate, in other words when

$$nr - n_p + \frac{n_d}{2} > 0 \tag{2}$$

where n is the number of frames transmitted, r is the rate of probes configured, n_p is the number of probes completed and n_d is the number of probes deferred. n_d is decremented and n_p incremented when a transmission report indicates the probe rate was used. Since probes might not be transmitted if the probe rate is below the current selected rate for normal transmission due to the structure of the retry chain, there is a mechanism to avoid bursts of probe frames by limiting the backlog of probe frames to less than twice the number of rates in use, by artificially increasing n_p . The frame counters n, n_p and n_d are reset to zero when n exceeds 10000. No two consecutive frames can be probes.

Rates to probe are chosen by round-robin from a randomly orderered static list to avoid random number generation in the transmit fast-path. Pseudo-random order was chosen because there are cases in which stepping through the rates in either sequential order is far worse; these are the same cases that defeat step up/step down algorithms. Avoiding expensive random number generation is also the reason for the rather complex probe frame selection mechanism. Probes are omitted if the rate to probe is already showing over 95% estimated success probability. The lowest rate in the supported rate set is never explicitly probed, since if it is ever needed the retry chain will provide sufficient probing.

5.3 Aging of Success Data

As frames are queued, the rate selection routine determines if a statistics update is required; this is done on the first transmission more than $\Delta t = 100$ ms after the previous statistics update. For each rate, the ratio of successes to attempts in the interval is calculated, and

⁸This sampling approach led to the algorithm's name; a minstrel wanders around, playing in various places, and staying longer at those places that pay better.

the recorded success probability is updated

$$p_R(t + \Delta t) = \begin{cases} (1 - w)p_R(t) + w \frac{n_{sR}}{n_{aR}} & \text{if } n_{aR} > 0 \\ p_R(t) & \text{if } n_{aR} = 0 \end{cases}$$
(3)

where n_{sR} is the number of successes at rate R in the interval between t and $t + \Delta t$, n_{aR} is the number of attempts at rate R in the interval, and w is the weight for the moving average. If n_{aR} is zero, the calculation is skipped, both to avoid division by zero and because, empirically, it is better to have old statistics than no statistics.

After updating probabilities, the throughput estimation, equation 1, is evaluated for each rate, and then the maximum throughput rate maxtp, the second highest throughput rate tp2 and the best probability rate maxp are stored for constructing retry chains. The lowest base rate in the current rate set⁹ is called lowest, and when probing there is a rate probe (see Table 1).

Due to the large number of retries at high rates, if such a rate suddenly ceases to work, the EWMA will fall very rapidly, and at the next update a different rate is nearly certain to be chosen. But because of the long retry chain, the packets that caused this update are nevertheless very likely to be delivered.

This update interval Δt combines with equation 3 to give a 1/e time constant τ for Minstrel's response to acknowledgments of

$$\tau = \Delta t \frac{1 - w}{w} \tag{4}$$

which is 33 ms for the default $\Delta t = 100$ ms and w = 0.75. Therefore Minstrel is very heavily weighted to the most recent update interval; if every attempt in an interval fails, the success probability estimate can fall to approximately 5% of its previous value¹⁰.

Since Wong, Lu, Yang, and Bharghavan [2006] showed that the mutual information of two frame successes decays over approximately 150 ms, an update interval rather shorter than 100 ms may improve the performance on rapidly fluctuating links, however it is also possible that the rather long retry chains account for this effectively. Hardware limitations make it difficult to reduce this interval, since transmission success reports have a lower priority than transmissions and therefore may be delayed, reducing the information that may accumulate in shorter time periods. Empirical investigation showed that there is little to be gained by modifying Δt and w,

and values that lead to a τ much longer than 100 ms or shorter than 20 ms lead to performance decreases.

5.4 Multi-rate Retry

Normal frame	Probe frame	Probe frame		
	$\mathtt{probe} > \mathtt{maxtp}$	${\tt probe} < {\tt maxtp}$		
maxtp	probe	maxtp		
tp2	maxtp	probe		
maxp	\mathtt{maxp}	maxp		
lowest	lowest	lowest		

Table 2: Multi-rate retry chains for normal data packets and probes. The columns are the order in which rates are placed in the retry chain. Retry counts are chosen from per-rate data from Table 1.

The Atheros radio cards we used for initially development of Minstrel support a multi-rate retry mechanism, as do many 802.11 radios. For each packet to be transmitted, there is a chain of four (rate, retry count) pairs. This complicates rate selection, while also permitting considerably better performance. Not all hardware supported in Linux is capable of multi-rate retry, so this selection mechanism is optional.

Multi-rate retry is supported in Minstrel by allocating a time budget for transmitting each frame and choosing a number of retries for each rate that fills the time allocated. For 802.11g, that budget is 24 ms, giving 6 ms for each stage of the retry chain, sufficient time for 10 retries at 54 Mbps, but only one at 1 Mbps. Separate calculations are done for RTS/CTS, CTS alone, and direct transmission, since the time taken for the extra exchanges must be accounted for. The exact calculation is outside the scope of Minstrel, it is implemented by requesting transmit time calculations from the hardware driver. Usually IEEE Std 802.11 [2007] or its revisions will specify time calculations, but Minstrel does support non-standard extensions provided the driver provides the calculation method.

The maximum number of retries attempted is therefore 33 for an 802.11a radio, or 31 for an 802.11bg radio, but may vary if non-default timings are used or there are hardware limits. The 24 ms bound was chosen as a compromise between allowing retries at the lowest rates supported by 802.11 hardware and the requirements of interactive applications such as VoIP and games.

Probe frames for rates where the present success probability estimate is less than 10% or greater than 95% are given half the normal number of retry attempts, to a maximum of two. This minimises the performance impact of probing, while allowing sufficient information into the table to adapt to sudden loss of a rate.

Rates are chosen in the order shown in Table 2, retry chain lengths are looked up in the per-rate data once the

⁹802.11 has some negotiation of allowed rates, so the current rate set is maintained per-neighbour. Minstrel is reset by mac80211 whenever the current rate set changes; this is harmless because the negotiation process normally completes within a few packets of the neighbour being discovered.

¹⁰This response is so fast that the Linux 802.11s implementors added detection of link failure to Minstrel so the 802.11s routing algorithm is very quickly triggered to route around failed links.

rates are chosen. Note that when the probe rate is lower than the maximum throughput rate, the probe rate comes second in the chain. This leads to the complication that many of these probes are not used in practice, as the best throughput rate has to fail all its retry attempts in order for the probe rate to be attempted. The Linux implementation of Minstrel limits the rate at which probes can be issued if there is a backlog of probing because of this issue.

When the hardware reports on completion of a transmission, the resulting retry chain report is examined to account for attempts and success at all rates at which frames were transmitted. Minstrel accounts successes to the rate that actually succeeded, regarding each attempt as an independant experiment on the capabilities of the channel. Completion reports may arrive at the rate control algorithm some time after the radio hardware has finished working on the frame; we have measured this latency to be more than 10 ms on Atheros hardware.

SampleRate accounts packet successes to the initial rate in a multi-rate retry chain, no matter which rate actually succeeded, and this causes a bias toward higher rates that is not supported by the evidence gathered during algorithm operation.

5.5 Status reporting

Minstrel publishes its statistics tables for each remote node through the Linux debugfs mechanism. This provides a useful diagnostic for the radio system and the state of the link to a remote system. Tables 3 and 4 present samples of the output¹¹.

The particular link shown in Table 3 is outdoor, over about 300 m range, using outdoor-specific timing adjustments, omnidirectional antennas and amplifiers. It is fairly typical of a fading and high-interference scenario. Scanning for networks using one of the antennas reveals more than 300 beacons visible on the operational channel (which is the quietest in the band). The link crosses a busy road in an urban area, so is undoubtedly affected by multipath and fast fading.

Given the amount of traffic shown on 36 and 48 Mbps, this link was running slower than is typical; unsurprising, given that it was raining at the time this sample was taken. It also shows one behaviour typical of Minstrel that is often surprising: the retry chain contains 11, 24, 5.5 and 1 Mbps, in that order. This link had been up for 26 days at the time the sample was taken, with rather light utilisation.

The link shown in Table 4 is a typical short-range indoor link with standards-compliant settings and 2 dBi antennas, sampled after 42 days of uptime. This node was acting as the client for a traffic generator for some

of that time, so those 563 million packets were mostly acknowledgements; the interface had received nearly a billion packets. Notice that Minstrel found a working high rate using only a relatively small number of packets. There has been some use of lower rates; while this is an indoor link, it is also a high-interference environment.

These tables are useful for determining if a link is usable or not; if all the ewma prob values are very low and yet have been probed more than a few times, there is no useful communication possible. One can watch the algorithm at work by repeatedly displaying the tables, which we use as an antenna siting tool.

6. TESTING RATE CONTROL

Rate control algorithms are difficult to test. In a practical test, a variety of scenarios should be constructed, selected to provoke the known issues. A practical rate control algorithm must be stable in the presence of any sources of packet loss, even those which cannot be adequately modeled, and must be stable when those sources of loss abruptly change. Finally, rate control algorithms must be able to deliver the initial signalling packets of practical networking protocols.

Since Minstrel, and indeed any rate algorithm that will deal with random packet loss, can achieve reasonable performance on even very poor links, driver bugs that cause packet loss on receive can easily be masked. This was the case for the 'ramping' bug, although the extreme round trip time variation due to that bug was readily apparent. It should also be noted that these testing processes also find issues with every other component of a wireless router or access point; we have found IBSS TSF desynchronisation bugs, IBSS merging bugs, DMA related issues in the platform support for several embedded development boards, routing code memory leaks and convergence problems, to mention but a few.

In every case, using an extra wireless node to capture packet traces and carefully studying them for correct rate algorithm operation, unexpected delays, spurious TCP retransmits and TCP stalls is a valuable and necessary exercise for debugging the rate algorithm and the rest of the MAC.

Testing on real hardware with real network protocols is required to adequately cover all the potential issues, especially ARP and connection exchanges for TCP and other connection-oriented protocols. Here we detail our standard tests for rate control (these are obviously not adequate to cover the entire system, although they do cover more than rate control alone).

First is the 'smoke test': run two nodes on a workbench separated by far enough to avoid front-end saturation. Normally this is around a meter, but high-power nodes intended for outdoor use should have their power turned down or be moved further apart. In this situation, with normal indoor interference levels (which may be

¹¹Output is extracted from a system running Minstrel
by cat /sys/kernel/debug/ieee80211/phy*/stations/
*/rc_stats

rat	е	throughput	ewma prob	this prob	this succ	/attemp	t success	attempts
	1	1.1	58.5	0.0	0(0)	3777	3983
	2	3.1	91.8	0.0	0(0)	2250	2341
P	5.5	6.5	99.9	100.0	1(1)	3622	3757
	6	5.3	54.1	0.0	0(0)	594	1518
	9	6.2	48.4	0.0	0(0)	1574	3640
T	11	8.8	99.9	100.0	1(1)	40377	41424
	12	8.3	53.2	0.0	0(0)	4134	8257
	18	7.2	37.9	0.0	0(0)	11569	22143
t	24	8.7	40.0	0.0	0(3)	38421	66339
	36	8.0	32.0	25.0	1(4)	115635	213476
	48	7.7	28.3	0.0	0(0)	188447	368039
	54	6.3	22.7	0.0	0(0)	68718	166525

Table 3: Sample Minstrel debugfs statistics table output for one outdoor node, approximately 300 m away from the reporting node. The rate column is the 802.11g rate. Letters to the left of the rate column show the current multi-rate retry chain; T is maxtp, t is tp2 and P is maxp. The throughput column is the most recent throughput estimate for that rate from Equation 1, in Mbps for readability. ewma prob is Equation 3 as a percentage, this prob is the success ration for the last update interval as a percentage, for readability. this succ/attempt are the successful and attempted trials during the interval preceding the last update. Finally success and attempts are the number of packets per rate that succeeded and were attempted, respectively, since startup.

rate	throughput	ewma prob	this prob	this succ	/atte	mpt success	attempts
1	0.7	81.3	100.0	0(0)	645	938
2	1.9	99.9	100.0	0(0)	11753	12602
5	.5 5.0	99.1	100.0	0(0)	11849	12662
11	8.3	87.4	50.0	0(0)	12328	13234
6	5.6	99.5	100.0	0(0)	11548	12455
9	8.4	99.6	100.0	0(0)	11675	12618
12	11.0	99.9	100.0	0(0)	12040	12983
18	14.7	90.5	100.0	0(0)	11879	12988
24	20.4	96.0	100.0	0(0)	12301	13595
36	26.6	87.4	100.0	0(0)	53171	58416
t 48	38.7	99.2	100.0	0(0)	1108136	1182045
T P 54	43.3	99.9	100.0	1(1)	562355519	583412540

Table 4: Sample Minstrel debugfs statistics table output for one indoor node. Same column definitions as Table 3.

hard to achieve in the offices of a company or lab that specialises in wireless), one expects to see high rates achieved reliably and TCP throughput approaching the limit of the MAC/PHY combination (about 22 Mbps for 802.11g, for example). Once and PID frequently give much lower performance than the best fixed rate in this simple scenario.

The second test requires a large obstacle; we use the shear bracing walls of an earthquake-rated reinforced concrete building for this test, however even in a lightly built structure this can be achieved by attaching some wire mesh to a partition wall. The idea here is to qualitatively examine the responsiveness of the algorithm. One node will be stationary for the test, the other should be movable (or at least its antenna should be on a sufficiently long cable). The movable node will be moved such that an obstruction comes between it and the stationary node, while repeatedly running backto-back short TCP transfers using a tool like iperf.

The expectation is that the TCP throughput will drop immediately as the obstacle intervenes, but that the transfers should not stall or create large numbers of consecutive TCP retransmits. When the movable node is moved back into clear view of the stationary node, the TCP throughput should increase very quickly (a few hundred ms), and again there should be no stalls or consecutive TCP retransmits. It is worthwhile to run the link for several days in clear view with traffic, then move into the shadow of the obstacle observing for TCP stalls, wait several days again and then move the node out. We have observed some algorithms responding quickly during the first hour of operation, and very gradually becoming unresponsive to changes (SampleRate is the best example of this behaviour). Other algorithms work fine with continous traffic, but become unresponsive after a period of idleness. Minstrel's response time is dependent on traffic, but not on time since startup

The third test is to graph throughput achieved versus

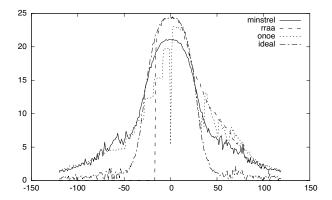


Figure 1: Data rate achieved (Mbps) for Minstrel, RRAA, Onoe, and 'Ideal' rate control algorithms versus position (m). Each trace is a simulation result from ns-3 using Nakagami propagation model and mobility, moving one 802.11a node from -120 m to 120 m on a track passing 1 m from a second node at 1 ms⁻¹. A full IP stack is installed on each node, the application is UDP echo transmitting 1024 byte payloads every $200~\mu s$, bandwidth includes IP and link level headers.

range, as in the simulation reports in Figures 1 and 2 below. As Minstrel does in Figure 1, a successful rate control algorithm should track the performance envelope established for manually selected fixed rates, and may even exceed that envelope in the presence of fading. During our assessment of rate algorithms for our mesh networks, all the other rate algorithms available to us failed this test in one scenario or another, either showing reduced range or reduced throughput. We found that Minstrel consistently matches the best manually fixed rate for every link we could test (especially outdoor links), whereas Onoe, PID and SampleRate frequently fall below, even to the extent of completely failing to establish connections.

The fourth test is stability; the algorithm should be left running for a long period of time (a few weeks, at a minimum) in a high-interference outdoor environment with traffic present on the network. We typically regard a wireless system as stable if it can run continuously for eight weeks in the presence of other networks and all the usual sources of interference in an urban area. Performance should stay within fairly tight bounds during such a test, although, of course, the daily cycle of the environment and the weather will have some impact. Minstrel is highly stable, we have run systems for months with no significant change in throughput, round-trip time, or responsiveness.

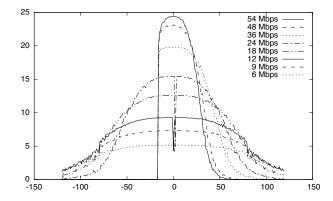


Figure 2: UDP echo data rate achieved (Mbps) for constant 802.11a rates versus position (m), same simulation configuration as Figure 1.

7. SIMULATION

While we developed Minstrel by testing on actual hardware, here we present some simulation results to demonstrate its behaviour in a more repeatable setting.

Minstrel and several other rate control algorithms have also been implemented in the ns-3 network simulator [Henderson et al. 2006, Henderson et al. 2009, Nguyen 2009]. We have verified by static analysis that the ns-3 implementation does correspond to the Linux kernel, although the code is in C++ and therefore looks rather different. For the cases presented in this paper, we configured the simulator with a Nakagami fading propagation model [Nakagami 1960], to give a channel similar to typical fading environments¹². Two nodes were constructed with YansWifi 802.11a radios in IBSS mode and full IPv4 stacks, including ARP. The simulator then moved one node from -120 m to 120 m position on a track passing 1 m from the other node at 1 ms⁻¹ while attempting to send 1024 byte UDP payloads every $200 \mu s$ to an echo server and recording the data rate actually transferred (including link layer headers)¹³.

The results for constant rates are presented in Figure 2, while the results for Onoe [Wong et al. 2006], Minstrel, RRAA [Wong et al. 2006], and 'Ideal' rate control algorithms are presented in Figure 1. SampleRate and PID

¹²The Nakagami parameters were ns-3 defaults, except $m_0 = 1.0$ and $m_1 = 0.75$, which increases the fading intensity substantially from default.

 $^{^{13}\}mathrm{This}$ application is approximately the behaviour of iperf -u -b 5120k -l 1024.

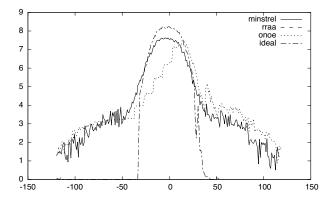


Figure 3: TCP data rate achieved (Mbps) for Minstrel, RRAA, Onoe, and 'Ideal' rate control algorithms versus position (m). Application is a one-way TCP transfer using 1024 byte socket writes at 12.5 Mbps, bandwidth reported for TCP frames containing data only, includes IP header but not link layer header, otherwise the same scenario as Figure 1. Note that RRAA transmitted no data.

are not simulated because there are no implementations of those algorithms in ns-3. The glitch near minimum distance appears to be a simulator artifact, although at 1 m range front-end saturation could be impairing actual radios.

At some of the higher fixed rates, ARP resolution does not complete until the nodes are quite close together. Disregarding the ARP issue, if a rate control algorithm is to match or exceed manually selected fixed rate in static situations and be responsive to changes or mobility, its results for this simulation should closely follow the envelope of the fixed rate curves.

Onoe is a step-up, step-down type rate control algorithm. 'Ideal' is a SNR based algorithm that assumes an ideal clear channel without interference, multipath or fading. As can be seen from Figure 1, neither Onoe nor 'Ideal' can consistently deliver good performance in this fading environment. Onoe lags the mobile scenario by 10 to 15 seconds, and the sawtooth pattern visible on the receding (right) side of the graph shows its unstable response to mobility. 'Ideal' performs quite well at short ranges, but fails to achieve significant throughput at long ranges, because it assumes a clear channel without fading or random loss. If a user were to reconfigure from 'Ideal' to Minstrel, a reasonable conclusion is that Minstrel increased the effective range of the radio system.

RRAA is an acknowledgement based step up/step down algorithm which estimates short-term frame loss, independent of rate, and uses a pre-calculated table of trigger loss rates to decide on stepping. It makes no use of multi-rate retry, but does adaptively turn on RTS to avoid some interference scenarios. RRAA can step

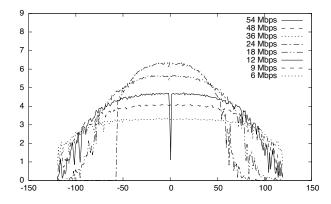


Figure 4: TCP data rate achieved (Mbps) for constant 802.11a rates versus position (m), same simulation configuration as Figure 3. Note that there is no data received at 36, 48 or 54 Mbps.

multiple rates in a single decision, and is responsive on extremely short timescales [Wong et al. 2006].

RRAA achieves no throughput until very close to the second node. Examining the packet trace for this run reveals the reason, which is that ARP replies are not being delivered at the 54 Mbps initial data rate chosen until the range reduces. This is a severe flaw in any rate control algorithm, as it completely prevents communication. We note that RRAA achieves approximately 15% better throughput than Minstrel at medium ranges once it has a connection, which suggests that it may deserve further investigation after fixing its robustness issues.

Minstrel, on the other hand, is consistently close to the envelope of the reference rates. At higher rates, there is some penalty for sampling, and also there is a small amount of training delay due to the mobile scenario. The 24 ms multi-rate retry chain is long enough that very short bursts of loss are unlikely to cause packet losses.

As was observed by Wong, Lu, Yang, and Bharghavan [2006], statistical sampling rate control algorithms suffer from training delay; it takes a relatively large number of transmissions and potentially several iterations of the statistics update for Minstrel to converge at a high rate. Although this is mitigated by probing in random order, the training process is still perceptible to the user; it can take up to approximately 500 ms even if there is sufficient traffic to sample the link. Training delay also imposes a significant penalty if a high rate suddenly ceases working; since the lower rates are unlikely to have accurate statistics in that situation, retraining proceeds almost from scratch. However, packet loss

visible to upper layers is quite low in that situation due to the construction of the retry chain, so Minstrel does achieve the goal of preventing TCP stalls wherever possible. Vutukuru, Balakrishnan, and Jamieson [2009] propose a rate adaptation mechanism called *SoftRate* that can adapt without training delay. Since it relies on protocol alterations to communicate physical layer information that is not exposed in commodity radio chipsets, *SoftRate* cannot be applied in the same way as Minstrel.

Figures 3 and 4 are results from the same simulator configuration, except that the application is a one-way TCP transfer using 1024 byte socket writes at 12.5 Mbps¹⁴. Since no cases actually transferred 12.5 Mbps, this will be link limited not application limited. In this configuration, the bandwidth reported is for IP packets containing TCP data delivered to the receiving node, and does not include link-layer headers. The issues some configurations displayed with ARP are magnified here by the TCP SYN exchange also failing, to the extent that some fixed rates and RRAA completely fail to deliver any data. Minstrel, on the other hand, establishes a connection immediately.

By simulating with the full IP stack, this simulation exercise has shown that there can be issues with rate algorithms that are not obvious using simulations confined to the link-layer. Initialising the rate algorithm such that ARP and initial transport protocol packets do not get delivered with high probability where that is possible is a serious flaw, and leads to the algorithm being unusable in practice. This simulation does not include background interference, or loss local to the radio as reported by Salvers, Striegel, and Poellabauer [2008], so it is clear that other issues may exist that are not shown (for example, on the outdoor link of Table 3, Onoe can not raise the transmit rate above 1 Mbps due to the interference level). As a result, these simulations are overly optimistic regarding the performance of the less robust rate control algorithms, particularly Onoe (which is very unreliable in practice).

8. FAILURES OF RATE ALGORITHMS

Rate algorithms can fail in various ways. Minstrel shows none of these failures, each of the other algorithms considered fails in at least one way.

Rate adaptation can cause ARP and TCP SYN failures; RRAA fails in this way in our simulation, Onoe and PID frequently fail this way in practice. Performance can be highly unstable; Onoe in particular can oscillate continuously, with dips to very low values. Algorithms can become unresponsive to environmental changes; the SampleRate implementation stops responding within hours. Slow response to changes can lead to poor perfor-

mance or connection failures; Onoe, SampleRate, and PID all fail this way. The round trip time can become very large due to buffering or head of line blocking because the rate algorithm uses too many retries at low rates; Onoe again fails, giving round trip times over a second in the presence of mild interference. Finally, if an algorithm drops too many consecutive frames, TCP connections can go in to long retransmit timeouts, and effectively stop; PID and Onoe will cause this sort of issue on marginal links.

The effective range of an 802.11 system is crucially dependant on the rate adaptation algorithm; 'Ideal', in the simulations above, halved the range of the system, and RRAA also reduced the effective range by being unable to connect with a weak signal.

We have observed these failure modes in commercial access points and the proprietary drivers and firmware for laptops and embedded devices. One cannot easily know the precise algorithms in use in many cases, but by capturing packets off the air it is possible to observe their behaviour. By now, most such devices err on the side of stability and robustness at the expense of low performance (usually by biasing choices toward low rates), which is of course better for the end user since the failure modes can be so catastrophic. However, bias towards low rates can hurt robustness in some situations.

Minstrel demonstrates that this trade is not necessary; it is possible to construct a robust, stable rate algorithm that also achieves near-optimal performance.

9. FUTURE WORK

While it seems likely to be possible to exceed the performance of Minstrel by about 15% on good links, we believe it to be very difficult to improve on the robustness of Minstrel on poor links. We have seen Minstrel achieve several Mbps of TCP throughput on links where no fixed rate will complete a large transfer. Therefore, future performance work on Minstrel should concentrate on adaptively reducing overhead on the best links, while retaining the robustness should the link deteriorate.

Minstrel has not been extensively tested on 802.11n hardware. Given the very large number of available rates and corresponding small distinction between the performances of adjacent rates, Minstrel may be further from optimal performance in this case, which may imply that adjustments to the probing schedule and throughput estimation equation can improve performance. Minstrel is known to be very robust on 802.11n systems.

Minstrel does not make use of the channel state feed-back available in some 802.11n systems, which may prove to improve performance. Similarly, approaches like SoftRate that use BER feedback appear to have value. However, the acknowledgement feedback mechanism of Minstrel appears to be vital to achieve robustness in

 $^{^{14}{\}rm This}$ application is the behaviour of iperf -b 12500k -s 1024.

the face of unpredictable errors (PLCP errors, hardware and software bugs, etc).

The constraints on rate adaptation imposed by 802.11, while ensuring interoperability, also reduce the robustness of the link. There is value in work on the 802.11 standard to extend rate adaptation to multicast and management frames, and potentially to improve the PLCP header.

10. CONCLUSION

Minstrel succeeds because it is responsive on an appropriate timescale given the constraints of commodity hardware, stable in the presence of any source of packet loss, delivers packets with very high probability, bounds the link-layer induced round trip time variability, optimises performance using a reasonable metric, and makes all its operational decisions on the basis of accumulated evidence.

Minstrel is a practical rate control algorithm for 802.11, and widely deployed due to being the default in mainline Linux and many distributions, for those radios on which it is supported. By making few assumptions and therefore responding appropriately to all sources of packet loss, Minstrel achieves good performance over poor links in the presence of interference where previous rate algorithms available for Linux mac80211 do not.

If Minstrel fails to achieve communication with a neighbour, it collects good evidence that no communication is possible. Conversely, if communication is at all possible, Minstrel will find rates to use that will achieve the best reliability and close to the best performance available.

11. REFERENCES

AGUAYO, D., BICKET, J., BISWAS, S., JUDD, G., AND MORRIS, R. 2004. Link-level measurements from an 802.11b mesh network. In *In SIGCOMM*. 121–132.

BICKET, J. C. 2005. Bit-rate selection in wireless networks. Tech. rep., Masters thesis, MIT.

Brivio, S. 2008. PID - linux wireless.

http://linuxwireless.org/en/developers/

Documentation/mac80211/RateControl/PID.

BRODSKY, M. Z. AND MORRIS, R. T. 2009. In defense of wireless carrier sense. In SIGCOMM '09: Proceedings of the ACM SIGCOMM 2009 conference on Data communication. ACM, New York, NY, USA, 147–158.

FIETKAU, F. 2008. mac80211: add the 'minstrel' rate control algorithm. Linux kernel commit

http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;

h=cccf129f820e431d84690729254a32f1709328fb.

Henderson, T., Spagnolo, P., and Kim, J. 2003. A wireless interface type for OSPF. In *Military*

Communications Conference, MILCOM. Vol. 2. IEEE, 1256–1261.

HENDERSON, T. R., ROY, S., FLOYD, S., AND RILEY, G. F. 2006. ns-3 project goals. In WNS2 '06: Proceeding from the 2006 workshop on ns-2: the IP network simulator. ACM, New York, NY, USA, 13+. HENDERSON, T. R., ROY, S., FLOYD, S., AND RILEY, G. F. 2009. ns-3-dev version 2009-11-12. http://www.nsnam.org/.

IEEE Std 802.11. 2007. IEEE standard for information technology — Telecommunications and information exchange between systems — Local and metropolitan area networks-specific requirements — Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. Leffler, S. and Renzmann, M. 2008. Madwifi version 0.9.4-current.

http://madwifi-project.org/wiki/Releases/0.9.4. Lukas, G. 2007. Ad-hoc mode queueing / latency problems. http://madwifi-project.org/ticket/1154. Nakagami, N. 1960. The m-distribution, a general formula for intensity distribution of rapid fading. In Statistical Methods in Radio Wave Propagation, W. G. Hoffman, Ed. Oxford, England: Pergamon. Nguyen, D. 2009. Minstrel implementation for ns-3. http://www.nsnam.org/wiki/index.php/GSOC2009AcceptedProjects.

Padhye, J., Firoiu, V., Towsley, D., and Kurose, J. 1998. Modeling tcp throughput: a simple model and its empirical validation. SIGCOMM Comput. Commun. Rev. 28, 4 (October), 303–314. Rodriguez, L. R. 2008. mac80211: make minstrel the default rate control algorithm. Linux kernel commit http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;

 $h=8eb41c93685318d177276d1819915571aca7ebb1. \\ RODRIGUEZ,\ L.\ R.\ 2009a.\ ath9k\ developer\ mailing\ list\ posting\ archived\ at$

https://lists.ath9k.org/pipermail/ath9k-devel/2009-November/002693.html.

RODRIGUEZ, L. R. 2009b. ath9k developer mailing list posting archived at

https://lists.ath9k.org/pipermail/ath9k-devel/2009-November/002696.html.

Salyers, D. C., Striegel, A. D., and Poellabauer, C. 2008. Wireless reliability: Rethinking 802.11 packet loss. In WOWMOM '08: Proceedings of the 2008 International Symposium on a World of Wireless, Mobile and Multimedia Networks. IEEE Computer Society, Washington, DC, USA, 1–4.

SMITHIES, D. 2008. minstrel - linux wireless. http://linuxwireless.org/en/developers/Documentation/mac80211/RateControl/minstrel.

Torvalds, L. 2009. Linux 2.6.29. Linux kernel commit

http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;

h=8e0ee43bc2c3e19db56a4adaa9a9b04ce885cd84. VUTUKURU, M., BALAKRISHNAN, H., AND JAMIESON, K. 2009. Cross-Layer Wireless Bit Rate Adaptation. In ACM SIGCOMM. Barcelona, Spain. Wong, S. H. Y., Lu, S., Yang, H., and BHARGHAVAN, V. 2006. Robust rate adaptation for 802.11 wireless networks. In Proceedings of the 12th annual international conference on Mobile computing and networking (MobiCom '06). Proceedings of the 12th annual international conference on Mobile computing and networking (MobiCom '06), 146–157.