Computer Networks 81 (2015) 289–307

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Rate control in the mac80211 framework: Overview, evaluation and improvements

Wei Yin [a,b], Peizhao Hu [c,*], Jadwiga Indulska [a,b]

[a] The University of Queensland, Australia
[b] NICTA, Australia
[c] Rochester Institute of Technology, USA

ARTICLE INFO

Article history:
Received 22 September 2014
Received in revised form 20 January 2015
Accepted 22 February 2015
Available online 5 March 2015

Keywords:
802.11 networks
MAC rate control
mac80211 framework

ABSTRACT

From version 2.6.22, the mac80211 framework has been incorporated as part of the stock kernel in Linux. This means millions of Linux based devices depend on this framework to provide wireless networking. The framework provides functionalities and interfaces for wireless device drivers (e.g., ath5k) to delegate common tasks to the kernel and to exchange information between physical and upper layers. One of these tasks is rate control at the MAC layer, which uses a *metric* to evaluate the channel conditions and an *algorithm* to select a transmission rate that achieves the best performance objective (e.g., max throughput) for the given conditions. To the best of our knowledge, a comprehensive analysis of this framework and its rate control mechanisms does not exist. This paper is the first comprehensive study on rate control mechanisms supported by the mac80211 framework, including overview, evaluation and potential improvements of these mechanisms. In addition to proposing a way to solve the oscillation problem in one of the mechanisms, we propose and evaluate an adaptive and low-cost solution to address collisions due to the hidden terminal problem that has not been considered by both mechanisms. The results show more than 40% improvement if the proposed solution is used when hidden terminals are present.

1. Introduction

From smart phones to wearable devices (e.g., Google glasses), more and more devices of our everyday life are becoming ubiquitous and mobile. Wireless networking is one of the enabling technologies that make this futuristic vision a reality. Although research on wireless networking has been around for decades and its adoption in consumer market has increased dramatically [1], wireless networking fells short in reliability compared to its wired counterpart due to the shared nature of the wireless medium.

In the shared wireless medium, channel conditions are unpredictable due to node mobility, channel fading and/or interference. As a result of the channel dynamics, wireless link quality may vary over time, which has impact on the performance of the networks. Transmission rate is one of the network parameters that determine how fast a node can send data onto the wireless medium. In principle, if link quality is good, higher rates can result in higher goodput and lower airtime. In contrast, when channel quality is poor, the use of higher rates will increase the number of retransmissions and the chance of packet drops.

Therefore, adaptively selecting an appropriate transmission rate (as supported by the IEEE 802.11 standards

* Corresponding author at: Rochester Institute of Technology, Computer Science, 102 Lomb Memorial Drive, Rochester, NY 14623, USA. Tel.: +1 (585) 475 4712.
E-mail address: ph@cs.rit.edu (P. Hu).

http://dx.doi.org/10.1016/j.comnet.2015.02.019
1389-1286/© 2015 Elsevier B.V. All rights reserved.

[2]) for the given channel quality becomes an important task to improve per-hop performance. This is the task of rate control at the MAC layer. In the past, rate control in Linux was implemented by individual drivers for each specific hardware, through the hardware abstraction layer (HAL) [3]. As an example, four mechanisms that are based on acknowledgement feedbacks (ACK based), including Onoe [4], AMRR [5], SampleRate [6] and Minstrel [7], were implemented in the Madwifi driver for the earlier version of Linux kernels. In [3], we evaluated these rate control mechanisms and showed that Minstrel has the best performance in the Madwifi driver.

Since kernel version 2.6.22, the mac80211 framework has emerged as part of the stock kernel, which provides an abstraction for the underlying hardware and implements crucial functionalities—therefore greatly simplifying the development of new drivers and network tools. The mac80211 framework is used by a number of radio drivers, including *ath5k*—a replacement of the *Madwifi* driver for the Atheros AR5000 series chipsets, *ath9k* for Atheros AR9000 series chipsets (for the 802.11n standard), *iwlwifi* for Intel wireless chips, and *b43* for Broadcom wireless chips. Rate control is one of the crucial functionalities that have been incorporated into the framework, which means all Linux based devices are using the same rate control mechanisms as provided by this framework. To the best of our knowledge, a comprehensive analysis and evaluation of this framework and its rate control mechanisms does not exist. Therefore, this paper will be an important contribution to fill the gap and provides insights and lesson learned to readers who might be working on this feature. In [8] we presented the initial performance evaluation of the two default rate control mechanisms: Minstrel and PID, and proposed a potential enhancement in PID (PIDE) that solves its incapability to converge to the best rate. In [9] we presented a new rate control mechanism—RCELC, which is based on our link capacity estimation metric. RCELC differs in various aspects from Minstrel and PID, and RCELC outperforms these two mechanisms as discussed in our paper. It should be noted that the focus of this paper is not RCELC, but the mac80211 framework and its supported rate control mechanisms. Also, none of our previous proposals have provided a detailed analysis of the mac80211 framework and addressed the issue of performance degradation in collision environments.

This paper presents the first comprehensive analysis and evaluation of MAC-layer rate control on the mac80211 framework. We incorporate the preliminary findings, from our earlier paper [8], to provide an in-depth analysis of the two supported rate control mechanisms on the framework. In addition, we present our work on investigating two common types of collision problems and their impact on rate control at the MAC-layer. As a resolution for these problems, we present a collision-aware rate control mechanism, which adaptively turns on RTS/CTS mechanism to achieve higher throughput. The evaluation confirms that the collision aware mechanism enhances throughput significantly.

We make the following contributions in this paper:

- Overview of the mac80211 framework and the rate control mechanisms supported by the framework.
- Comprehensive evaluation of the two rate control mechanisms: Minstrel and PID.
- Analysis of issues affecting the PID performance and how the mechanism could be improved.
- Improvement by making Minstrel and PID collision-aware.
- Discussion on design guidelines for developing new rate control mechanisms.

The remainder of the paper is organised as follows. In Section 2 we briefly describe the mac80211 framework and two rate control mechanisms in the framework. The performance comparison of Minstrel and PID carried out in a conducted testbed for repeatable experiments is described in Section 3. The PID enhancement is discussed in Section 4. Here, we discuss PID shortcomings, present a PID enhancement (PIDE), and evaluate PIDE with controllable and over-the-air experiments. In Section 5, we discuss the collision problem in wireless networks and present the collision aware mechanism. The evaluation of the mechanism is also carried out and the results show its significant improvement. The related work is presented in Section 6. The paper concludes in Section 7.

2. Rate control on mac80211

In this section we briefly describe the mac80211 framework and two rate control mechanisms supported by this framework.

2.1. Linux 802.11 wireless stack

More and more wireless radio card manufacturers adopt the *SoftMAC* approach, which allows finer control of frame management to be done in software. This is in contrast to the *FullMAC* approach, which leaves all the control of the MAC layer functions to the card hardware/firmware [10]. The mac80211 framework has been introduced and integrated into the recent Linux kernels. This SoftMAC approach framework provides common MAC-layer functionalities for frame management (e.g., frame generation and parsing according to the 802.11 format) and control management operations related to the IEEE802.11 standard, rather than relying on the *ad hoc* implementation provided by individual wireless drivers.

Fig. 1 shows the Linux 802.11 wireless stack. The mac80211 framework is integrated into the Linux kernel

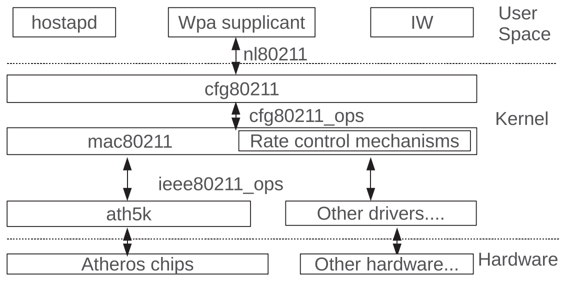


Fig. 1. Linux 802.11 wireless stack.

as a wireless subsystem. It provides configuration and management of the wireless radio and its data transmission through the user-space tools (e.g., the wireless utility daemon, *iw*). Along with the framework another component *cfg80211* was introduced to handle operations related to configuring the underlying hardware, via the *cfg80211_ops* interface. The communication between user-space tools and cfg80211 is implemented using netlink socket, *nl80211*. Between the mac80211 framework and physical devices there are individual drivers (e.g., ath5k), which implement common functionalities for hardware control, as defined by the framework in the *ieee80211_ops* interface.

The mac80211 framework acts as a middleware between the physical hardware and user-space tools, providing an abstraction for wireless devices that have different specifications. In addition, the framework provides common functionalities for individual drivers, which greatly simplifies the task of writing new drivers for those SoftMAC devices.

In addition to frame management, the mac80211 framework provides mechanisms for MAC-layer rate control. Individual drivers have an option to use the rate control mechanism provided by the framework, or to define their own rate control mechanisms and integrate them with the framework through an interface (*rate_control_ops* defined in the *rc80211* module). Typically, there are two types of MAC-layer rate control mechanisms: (i) *Per-frame* rate control, which usually has a dedicated processor to handle frame transmissions and receptions. It sets the rate for an outgoing frame and waits for it to be transmitted, and then gathers the status regarding the transmission in order to work out the best rate for the next frame. As specified in the IEEE 802.11 standard [2], the minimum time between two consecutive frame transmissions is at least a DIFS time (28 μs for 802.11g networks). In Linux, frame transmissions are managed using interrupt scheduling. There is no guarantee that an interrupt will be served within minimum latency. Therefore, rate control mechanisms developed for Linux are based on a *rate adaptation period* (RAP). (ii) A RAP based approach defines a window in which a transmission rate is specified. The performance of the selected rate within this RAP is averaged and is then used to determine the best rate for the next RAP.

At the core of every rate control mechanism there are two fundamental components: (i) A *metric* used to estimate the performance of different rates under current link quality. For example, throughput and frame loss ratio (FLR) are two commonly used metrics in rate control; the typical objectives when using these metrics are selecting a rate that can maximise the achieved throughput or maintain the FLR below a predefined threshold. To calculate these metrics, a rate control mechanism can access the transmission statistics at the physical-layer through interfaces (*ieee80211_ops*) defined by the mac80211 framework. (ii) An *algorithm* that defines strategies to select a transmission rate appropriate for the given channel conditions, with minimum delay. Prolonging the time spent on selecting an inappropriate transmission rate will degrade the performance.

When a frame is passed to the MAC-layer, the selected rate control mechanism will specify the transmission rates to be used (or in addition the retry count if multi-rates retry is supported) in the *Control Block* of the *sk_buff* buffer. After the transmission is completed, the driver then updates the *Control Block* with the transmission status. Different chipsets may report different set of statistics. For example, for the Atheros AR5212 chipset, the ath5k driver reports the signal strength for the received acknowledgement, attempted rates and their respective retry counts. Together with the information such as the number of frames sent/received and other physical-layer parameters, various link quality metrics can be calculated, including frame loss ratio, estimated link throughput, and SNR.

In the mac80211 framework, there are two rate control mechanisms: *Minstrel*, which tries to select a rate with maximum estimated achievable throughput, and *PID*, which selects a rate that aims to maintain FLR to be below a predefined FLR threshold.

2.2. Minstrel

Minstrel [7] is the only rate control mechanism that has been ported from the Madwifi driver onto the mac80211 framework. It outperforms the other mechanisms (i.e., Sample, Onoe and AMRR) available on the Madwifi driver [3].

To determine the appropriate transmission rate for the given channel conditions, Minstrel estimates the maximum achievable throughput (*TP*) of each rate and selects a rate that is capable to achieve the maximum throughput. The throughput for the given channel conditions is calculated as

$$TP = P_{new} * \left(\frac{one\_second}{T_{tx\_perfect}}\right) \tag{1}$$

where P_{new} is the weighted probability of success for the current RAP window that is to be used by the rate selection process and is computed in Eq. (2), and $T_{tx\_perfect}$ is the time required to successfully deliver a frame in perfect channel conditions; therefore, $\frac{one\_second}{T_{tx\_perfect}}$ is the number of packets that can be transmitted in one second window under the perfect channel conditions.

For every interval of 100 ms (default RAP length), Minstrel measures the statistics of frame delivery and computes P_{new} using the Exponential Weighted Moving Average (EWMA), which controls the balance of influence of both old and new packet delivery statistics.

$$P_{new} = (1 - \alpha) * P_{this\_interval} + \alpha * P_{previous} \tag{2}$$

where $P_{this\_interval}$ and $P_{previous}$ represent the probability of success for the interval before rate selection and for the last interval respectively. By default the smoothing factor α is set to 75%, which means historical throughput measurements have more weight on new rate selection.

According to the mac80211 framework source code within the 2.6.35 Linux-wireless kernel, Minstrel calculates the perfect transmission time, $T_{tx\_perfect}$, as

$$T_{tx\_perfect} = SIFS + T_{Frame} \tag{3}$$

where *SIFS* is a type of inter-frame spacing defined by the IEEE 802.11 standard [2]; T_{Frame} is the transmission time of the data frame, and it is calculated as

$$T_{Frame} = Ceil((16 + 8 * LEN + 6)/NDBPS) * T_{SYM} + T_{Preamble} + T_{Signal} \tag{4}$$

where T_{SYM} is the symbol interval (defaults to 4 µs for 20 MHz channel spacing), $T_{Preamble}$ is the PLCP header preamble duration (defaults to 16 µs), and T_{Signal} is the duration of the SINGAL BPSK-OFDM symbol (defaults to 4 µs). These coefficients are predefined in the IEEE 802.11 standard [2]; *LEN* is the frame size (fixed at 1200 byte in Minstrel), and *NDBPS* is calculated as $k * r$ where k is a coefficient and r is the bit rate. We can derive from the above equations that the Minstrel's throughput estimation depends on two parameters, the probability of success to deliver a frame and the bit rate it uses.

To account for retransmissions the Minstrel uses a Multi-Rate Retry (MRR) chain, which proposes four candidate rates (i.e., $r0, r1, r2$ and $r3$) to attempt in case re-transmissions are necessary. Each of these candidate rates has a corresponding counter (i.e., $c0, c1, c2$, and $c3$), which specifies the maximum number of retries for each candidate rate. For example, the algorithm will first attempt to use rate $r0$ for transmission; if the transmission fails $c0$ times, then rate $r1$ is attempted for re-transmission; in the case when rate $r1$ also fails $c1$ times, then rate $r2$ and $r3$ are attempted in the same manner until the base rate is reached.

To determine the optimal rate for a given channel condition, Minstrel dedicates 10% of its traffic to probe the performance statistics of other rates by randomly selecting a rate (as *Lookaround rate*) that is not currently in use. For this 10% of data traffic, as shown in Table 1, the retry preferences are the best throughput rate (BTR), the random rate (RR), the best probability rate (BPR) and the base rate (BR) if the randomly selected rate (*RR*) is lower than the current best throughput rate (*BTR*); otherwise they are the random rate, the best throughput rate, the best probability rate, and the base rate. For the other 90% of traffic (*normal packets*), the retry preferences are the best throughput rate, next best throughput rate (NBTR), the best probability rate and the base rate.

2.3. PID

The core of the PID (proportional–integral–derivative) [11] rate control mechanism is a proportional integral derivative (PID) controller. In essence, the controller is a control loop feedback mechanism that tries to minimise the difference (i.e., *error* in control system's term) of the current and target frame loss ratio (i.e., $FLR_{current}$ and FLR_{target} respectively) as a result of switching to a new transmission rate. By default, the FLR_{target} is set to 14% for all rates.

To determine the appropriate transmission rate, the controller computes an adjustment value, *adj*, as follow.

$$adj = \gamma * (1 + sharpening) * (e_{current} - e_{last}) + \alpha * e_{current} + \beta * e_{avg} \tag{5}$$

where $e_{current}$ is the current error, and its value is calculated as $FLR_{target} - FLR_{current}$. e_{avg} is the average of recent errors, while e_{last} is the last error. In addition, there are four tuneable parameters. *sharpening* is a smoothing factor (non-zero when fast response is needed), whereas α, β and γ are the corresponding *proportional*, *integral* and *derivative* coefficients. In all our evaluations, we use the default values for these coefficients. Although we can adjust these coefficients to change the behaviour of the PID mechanism, we will illustrate later that the oscillation problem of PID (as discussed in Section 3) lies in the algorithm rather than the selection of coefficient values.

Using Eq. (5) PID computes the adjustment value, *adj*, at the end of each rate adaptation period and decides on whether to switch to a new transmission rate. When *adj* is positive, the new rate R_{new} is set to the highest rate, in the range of $R_{current} \leqslant R_{new} \leqslant (R_{current} + adj)$, and its error (i.e., the difference between the target and respective frame loss ratio) is no more than the error of the current rate, $R_{current}$. When *adj* is negative, the new rate R_{new} is set to the lowest rate, in the range of $(R_{current} + adj) \leqslant R_{new} \leqslant R_{current}$, and its error is no more than the error of the current rate. No rate change is required, if *adj* is equal to zero.

Table 1
Retry preferences.

Attempt	Lookaround rate		Normal rate
	RR < BTR	*RR > BTR*	
$r0$	BTR	RR	BTR
$r1$	RR	BTR	NBTR
$r2$	BPR	BPR	BPR
$r3$	BR	BR	BR

3. Evaluation

In the last section, we explained the principle of two rate control mechanisms, PID and Minstrel. In this section, we discuss the evaluation of them on a controllable experimental platform and these results are also verified in real environment settings.

3.1. Conducted experiments setup

To ensure consistency of the evaluation results, we put together a conducted experiment setup, as shown in Fig. 2. We connect a traffic source to a traffic sink using co-axial cables and vary the link quality/channel conditions using a variable attenuator (Vaunix LabBrick LDA-602). The wireless signals are transmitted along the co-axial cables (not over the air by antennas). In addition experiment nodes are enclosed within RF shielding boxes (JRE 4400), which ensures 85 dB of isolation to external sources. Therefore, we argue that the platform can produce repeatable experiments, i.e., the rate control mechanisms are compared for the same parameters (e.g., path loss) and offered load. Although there may exist work on the performance of Minstrel and PID using over the air testbeds, with this

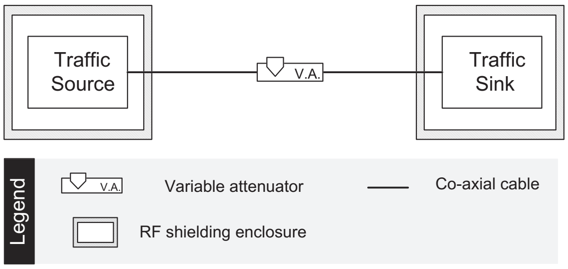


Fig. 2. Experiment setup.

conducted testbed our paper will be the first to present a rigorous and systematic study of these rate control mechanisms. It is almost impossible to produce repeatable experiments with over-the-air testbeds. It should be noted that during the experiments, the attenuation value varies according to the specifications defined in each evaluation scenario.

Each traffic source and traffic sink runs on a single board computer with the Wistron CM9 Atheros wireless card. Each computer runs the Linux operating system (with kernel version 2.6.35 and the corresponding mac80211 framework). The debugfs system for Minstrel and PID are modified to allow access to debug information (e.g., the selected rate). The transmission power of each node is set to 16 dBm. With the operation range of the variable attenuator (0–63 dB), a full range of link qualities is possible (i.e., minimal attenuation corresponds to full throughput and maximum attenuation corresponds to a disconnected link).

All experiments are performed using IEEE 802.11a. This has the benefit that all transmission rates use the same family of modulation and coding methods. After configuration, each setup is verified manually using various tools. All transmitted data packets are captured at the receiver using a packet sniffing tool (tcpdump). Each measurement contains information regarding the achievable throughput and the number of successful packets at each rate. A parser library (Banjax) is used to analyse the measurement files. The traffic is generated with UDP flows and the packet size is set to the iperf default size. Experiments using TCP packets are not discussed in this paper, as congestion control in TCP makes it difficult to distinguish whether the poor throughput is due to the congestion control mechanism or rate control at the MAC layer.

3.2. Conducted experiments: scenarios and results

To compare the performance of the two rate control mechanisms under different channel conditions, we define four representative scenarios: static channel, fading channel, progressive increase/decrease of channel quality and a sudden change in channel quality.

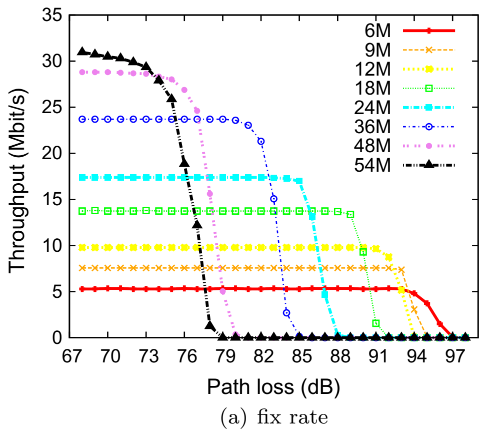
3.2.1. Static channel conditions

Under static channel conditions the rate control mechanisms should converge at an optimal rate and minimise hopping between different rates. For this scenario we fixed the effective path loss between two traffic nodes for a period of time and observed the achieved throughput and rate selection in relation to the given effective path loss. During each experiment the transmission power of the traffic source is set to 16 dBm, while iperf offered load is set to saturating the link. Results are presented in Fig. 3(b).

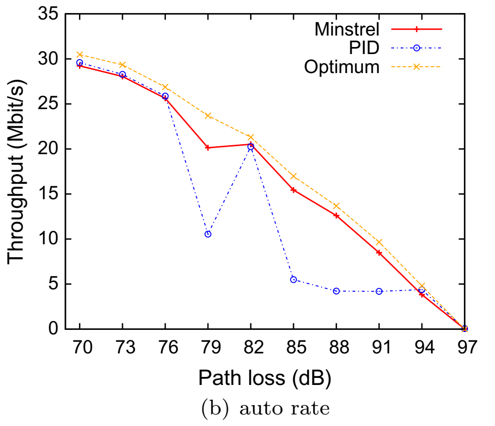
To show how well the two mechanisms perform, we also carried out experiments to determine the absolute *optimum* throughput (as a benchmark for the comparison) that can be achieved under the given channel conditions, using fixed rates. Fig. 3(a) shows the performance (for different path loss) of all fixed rates supported by the IEEE 802.11a standard. We define the optimum throughput as the envelop of the highest throughput achieved by fixed rates at different path loss.

By plotting the optimum throughput against the performance of the two mechanisms, Fig. 3(b) shows that Minstrel performs much better than PID, in some cases achieving nearly the optimum throughput.

As shown in Fig. 3(b), PID performs well in extreme channel conditions in which either the highest rate of 54 Mbit/s or the lowest rate achieves the best throughput. However, in other channel conditions (e.g., 79, 85, 88 and 91 dB) PID performs poorly. To investigate this problem,



(a) fix rate



(b) auto rate

Fig. 3. Fixed and auto rate performance in static channels.

we record the rate selection history for path loss at 85 dB and present it in Fig. 4. From this repeating pattern, we can see that PID tends to select a higher rate than the channel is capable of supporting; this results in higher frame loss ratio (FLR) and PID then falls back to the base rate and gradually increases the rate again. By analysing Fig. 3(a) and 4, we discover that this oscillation in rate selection is the result of the FLR of a selected rate becoming greater than the target FLR of 14% defined in PID; thus PID tends to increase rate. The oscillation occurs whenever a rate reaches the path loss regions (e.g., between 82 and 84 dB for 36 Mbit/s), as shown in Fig. 3(a), where the rate's throughput experiences sharp fall. In these path loss regions, it is likely that the FLR is above the defined target FLR, which causes PID to drop rate. This is a problem in rate control mechanisms if they switch to a higher rate, even if the channel conditions cannot support it.

3.2.2. Responsiveness of Minstrel and PID

Responsiveness is one of key aspects of a rate control mechanism; that is, how fast a rate control mechanism can select the optimal rate for the given channel conditions. To create such a scenario, we start the experiment with one path loss value; after a period of time, we either increase or decrease the path loss value to emulate sudden channel quality changes. From the point when attenuation is changed, we measure the time it takes for rate control mechanisms to converge on an appropriate rate for the new channel conditions.

Fig. 5 shows traces for these experiments. In the case when the channel quality suddenly increases (i.e., decrease in path loss) as shown in Fig. 5(a), we notice that Minstrel will take approximately 500 ms to stabilise at 54 Mbit/s for the new channel conditions. For PID, we again observe the repeating pattern as before. Obviously PID does not converge at a rate. While it may look that PID converges at 54 Mbit/s at around 5 s, 54 Mbit/s is in fact already the highest rate supported by the IEEE 802.11a standard; otherwise, the repeating pattern will continue. This is a fundamental issue of the PID rate selection mechanism—it fails to verify whether the higher rate should be used. As expected, we see a similar repeating pattern for PID in the case when channel quality decreases, as shown in Fig. 5(b). On the other hand, Minstrel takes around 600 ms to converge at 24 Mbit/s after the channel quality changes.

3.2.3. Channel quality progressive increase/decrease

In addition to channel fading we create a scenario in which the channel quality progressively increases/decreases to emulate two wireless nodes moving toward/away from each other. Changes in channel quality are controlled by varying the attenuator.

Fig. 6 shows that in both cases Minstrel outperforms PID. We argue that the oscillation in rate selection is the cause of such poor performance in PID. Another observation from the results is that Minstrel experiences about seven significant throughput drops when channel quality linearly decreases, as shown in Fig. 6(b). After a detailed analysis, we discover that the path loss values of these throughput drops correlate exactly with the path loss regions where the throughput of a fixed rate drops rapidly, as shown in Fig. 3(a). As an example, for the drop between the time 32 and 34 s, the path loss value increases from 82 to 84 dB. At this path loss, there should be a transition in

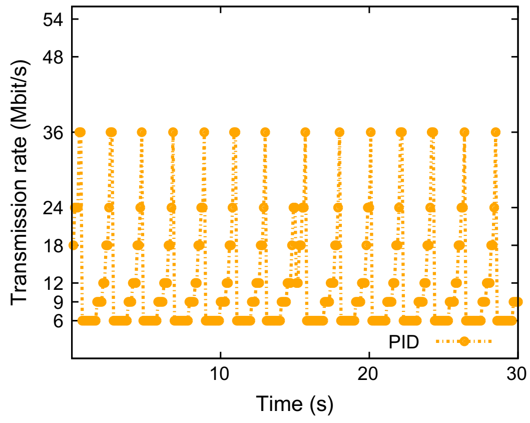
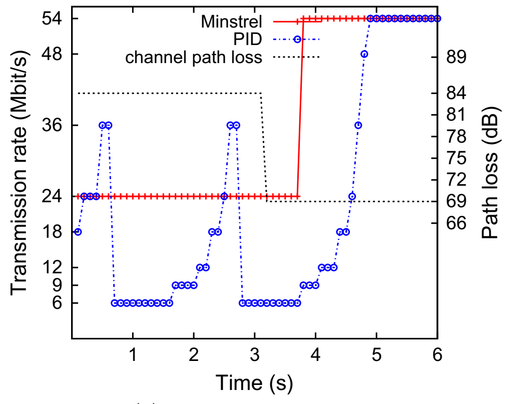
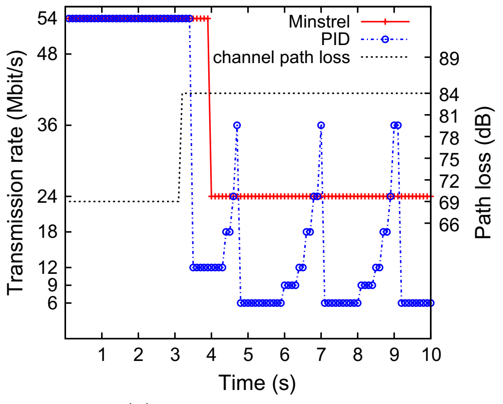


Fig. 4. Repeating pattern in rate selection in PID.

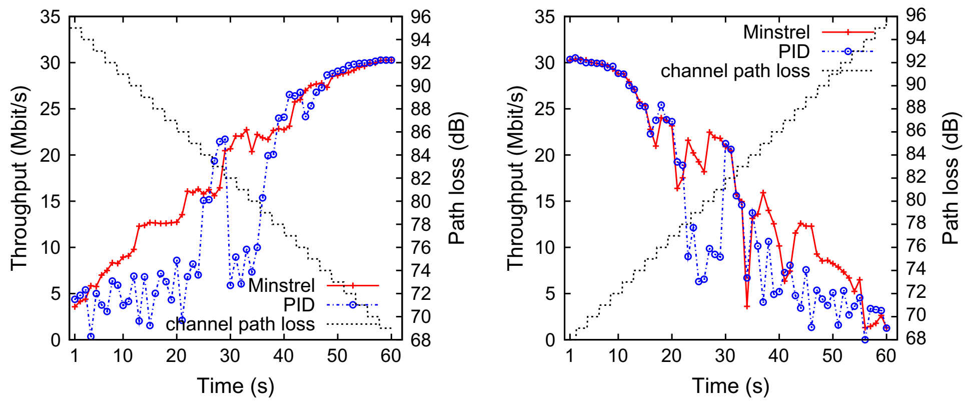


(a) Decrease attenuation



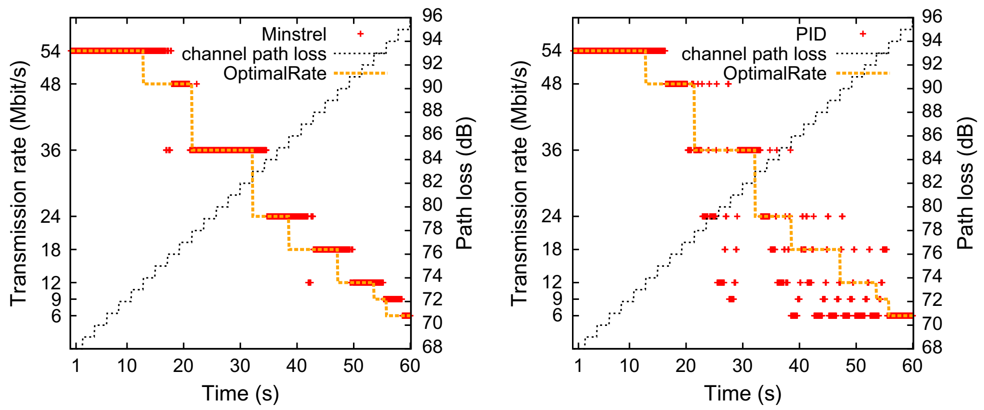
(b) Increase attenuation

Fig. 5. Rate selection as channel quality suddenly changes.



(a) Quality linearly increase (b) Quality linearly decrease

Fig. 6. Throughput as channel quality changes linearly.



(a) Minstrel rate selection (b) PID rate selection

Fig. 7. History of rate selection as channel quality decreases.

rate selection, from 36 to 24 Mbit/s. However, for Minstrel a great portion of the outgoing packets are still sent at 36 Mbit/s, rather than using the optimal rate, as shown in the history of rate selection when channel quality decreases (in Fig. 7(a)); some of them will fail and retry using lower rates, hence resulting in poor throughput. One reason for why Minstrel still selects 36 Mbit/s after the channel quality changed is due to the way Minstrel calculates its metric. As discussed in Section 2, the smoothing factor α (in Eq. (2)) defaults to 75% to increase the weight on history measurements in its calculation. Before the channel changes, the rate of 36 Mbit/s achieves highest throughput. It will take time for the 36 Mbit/s measurement to have no impact on the rate selection. A comprehensive study on the impact of α value on the Minstrel performance (on the Madwifi driver) is presented in [12].

3.2.4. Dynamic channel conditions

To study the performance of the two rate control mechanisms under varying channel conditions, we create a Rayleigh fading [13] scenario to emulate the effect of signal propagation in the wireless environments. The probability density function [13] of the received signal power is described as

$$p(\gamma) = \frac{\gamma}{\sigma^2} e^{-\frac{\gamma^2}{2\sigma^2}} (0 \leqslant \gamma \leqslant \infty) \qquad (6)$$

where r is the envelope amplitude of the received signal, and $2\sigma^2$ is the pre-detection mean power of the multipath signal. The Rayleigh distribution is controlled by σ, which is the root mean square value of the received signal.

Using the Rayleigh model, we generate 60,000 samples of attenuation values which are to be played-back on our programmable attenuator. The average path loss of the model is set to 85 dB. We vary the coherence time from 1 to 100 ms. The coherence time reflects how fast the channel changes in the Rayleigh channel fading model, which refers to the time period during which the channel remains constant. In [14], Camp and Knightly discuss that typical coherence time in residential urban area caused by passing vehicles is roughly between 15 and 100 ms. Fig. 8 shows that Minstrel outperforms PID by about 70%. We believe that the poor performance of PID is the result of the oscillation in rate selection.

3.3. Over-the-air experiments

In the last section, we discussed the evaluation of both rate control mechanisms in a conducted platform in which all RF signals are sent over coaxial cables rather than over-the-air. The advantage is that we have a full control of the experiments—the experiments are repeatable; hence we can analyse the cause of a particular observation in the measurements. To validate these conducted evaluation results, we put together an over-the-air setup (i.e., RF signals are sent via antenna) and present our findings from two experiment configurations.

Fig. 9 shows the random placement of four senders (i.e., S1, S2, S3, and S4), which send saturated traffic (one at a time) to the receiver (i.e., R) in our office environment. With this five nodes setup, we perform two experiments: *Semi-controlled* configurations in which we select a wireless channel and experiment time that have minimum level of external interference; and *In-the-wild* configurations in which we perform our experiments using a wireless channel that is shared with wireless access points around the office and at random office hours. The latter experiments will potentially test a rate control mechanism's ability to adapt to uncontrolled (but realistic) channel quality changes.

3.3.1. Semi-controlled configurations

The aim of this over-the-air experiment is to recreate the static channel scenario we have in the conducted experiment setup. We want to validate our findings in a real environment; that is, RF signals are sent via the antenna and from senders at different random locations (hence different propagation delays and different path loss to the receiver). To ease our analysis, we first run our experiments at midnight over the weekends and use a wireless channel that is not shared with other wireless access points in the vicinity. Each experiment runs for one hour, and the measurements are averaged.

Fig. 10(a) presents the throughput results for both mechanisms measured from different locations (refer to Fig. 9) and shows that Minstrel clearly outperforms PID at all locations. These experiments confirm the instability of rate selection in PID. Usually, performance in the semi-controlled environment should be higher than that in the wild environment, since minimal interference is introduced in the semi-controlled environment. However we notice that it is not the case for some locations, e.g., S4. This is due to frequency selective fading [15]. Same node pair using different communication frequencies results in different throughput.

3.3.2. In-the-wild configurations

In the *In-the-wild* experiments, we perform random tests during random office hour; therefore there will be people moving around in the office (or even using the microwave at a location between these nodes). In addition, we use a wireless channel that is shared with four university's wireless access points around the office; in terms of traffic pattern, they are completely out of our control. We run each experiment for one hour and average the measurements.

Fig. 10(b) shows that Minstrel outperforms PID. Due to the complexity involved in modelling the exact channel changes, we cannot provide the most definitive reason for the performance results. However, we strongly believe that the instability in rate selection will be one of the significant drawbacks, which causes poor performance for PID.

4. PID improvement

From previous experiments we can conclude that Minstrel outperforms PID. In this section, we discuss the potential reasons for the PID's poor performance, present a way to improve PID, and show that this PID enhancement significantly improves its throughput.

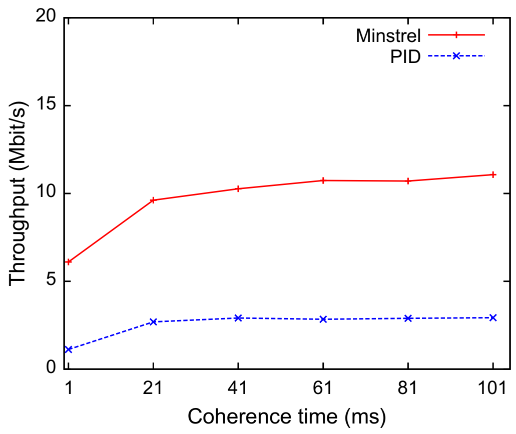


Fig. 8. Performance in Rayleigh model.

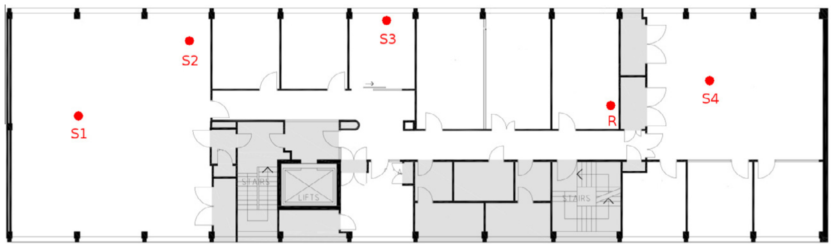


Fig. 9. Experimental floor plan.

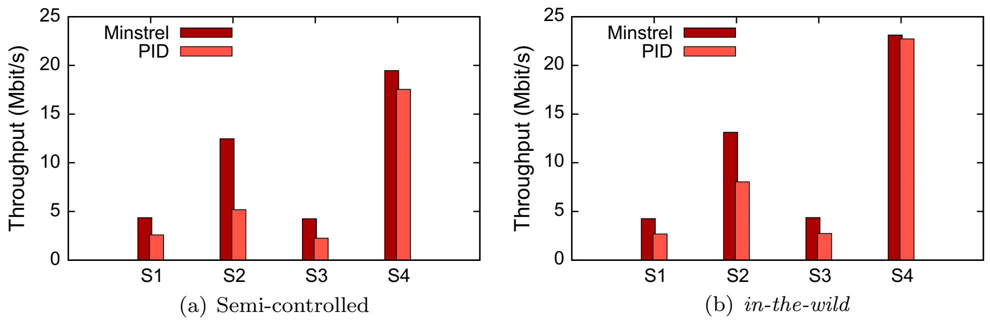


Fig. 10. Over the air performance.

4.1. Issues in PID

The reason that PID achieves low throughput even in static channels (as shown in Fig. 3(b)) is due to the oscillation in rate selection, which causes it to fail to converge to the optimal throughput rate, as shown in Fig. 4. As discussed in Section 2, the idea of PID is based on control system's principle of a feedback loop. That is, defining a target frame loss ratio (FLR)—e.g., the target FLR in PID is fixed at 14%—and let the rate control mechanism to adapt its rate to meet that target by minimising the difference between the current FLR and the target FLR. It is very straightforward; such that PID drops the sending rate when FLR is greater than 14%, while increases the sending rate when FLR is less than the target FLR.

During the adaptation process PID will increase the rate whenever the current FLR is below the target threshold, regardless of the current channel conditions which may not be sufficient to support the higher rate. The consequence of this is that the FLR of the higher rate is higher than the target threshold and this causes PID to drop the rate again. Hence, this results in oscillation in rate selection.

One solution to this problem is to implement a verification mechanism that probes the achieved throughput of the proposed rate compared to the current throughput. The goal is to maximise throughput therefore if the proposed rate achieves higher throughput than the current sending rate the algorithm should select the proposed rate. Otherwise, the rate adaptation requests should be ignored. The same rule applies for requests either to increase rate or to decrease rate.

4.2. The making of PIDE

In our enhancement of PID (PIDE), the rate control mechanism uses information passed by the PID to detect requests on rate adaptation at the end of each rate adaptation period. When a new rate is proposed (at the end of the last adaptation period) and the proposed rate has not been verified, PIDE then sends n frames (e.g., current implementation uses three frames) using the proposed rate in the current rate adaptation period. Other frames within the adaptation period will continue to use the current rate. Each frame is associated with a status that records a number of statistics regarding the transmission (e.g., retry count). PIDE collects these statistics and maintains a table of performance information for each rate. At the end of the current adaptation period, PIDE compares the achieved throughput to decide whether to use the proposed rate in the next rate adaptation period. The achieved throughput tp is calculated as

$$tp = (1 - \mathrm{FLR}) * (1s/T) \tag{7}$$

where FLR is the frame loss ratio, $1s$ is one second, and T is calculated as shown in Eq. (8). T_{DATA} and T_{ACK} are the respective transmission times of the DATA and ACK frames calculated based on Eq. (4) according to the IEEE standard [2]. BO is the backoff time which depends on the contention window size.

$$T = DIFS + BO + T_{DATA} + SIFS + T_{ACK} \tag{8}$$

In addition to the verification mechanism, we also modify the way frame loss ratio is calculated in PID. When calculating the FLR, PID counts multiple failures of frame transmission attempts as one failed frame; this potentially underestimates the frame loss ratio.

4.3. Evaluation of PIDE

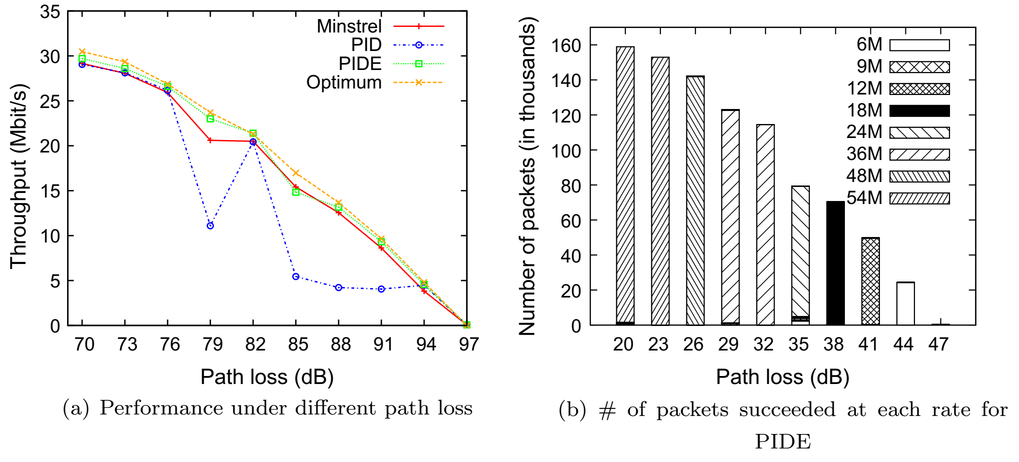
To show the effectiveness of our proposed enhancement, we revisit the same set of conducted and over-the-air experiments, as discussed in Section 3. Due to the space limitation, we only discuss some of them in the paper.

4.3.1. Static channel conditions

Fig. 11(a) shows that PIDE achieves much better performance than PID when the channel conditions are relatively static; on some occasions, PIDE even outperforms Minstrel and achieves throughput that is very close to the optimum. When looking into the history of rate selection as in Fig. 11(b), it is very obvious that PIDE solves the instability problem in PID. Rather than hoping around between different rates, PIDE is able to quickly converge at rates that can achieve high throughput and the maximum throughput gain over PID is around 200% as shown in Fig. 11(a).

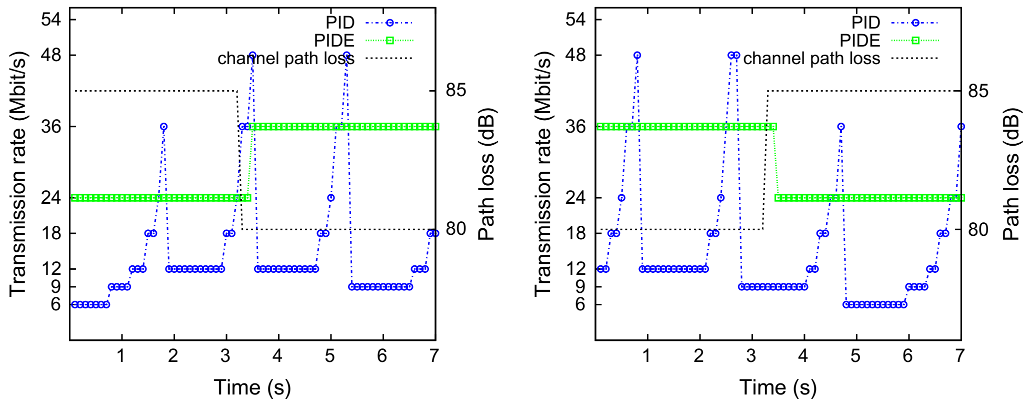
4.3.2. Responsiveness to sudden channel quality changes

In the experiments to test the responsiveness of PIDE due to sudden channel quality changes, PIDE shows significant improvement over PID, as it converges at an optimal rate in the matter of around 200 ms in both cases when



(a) Performance under different path loss (b) # of packets succeeded at each rate for PIDE

Fig. 11. Performance in static channel.



(a) Rate selection of PIDE. (b) Rate selection of PID.

Fig. 12. Rate selection as channel suddenly changes.

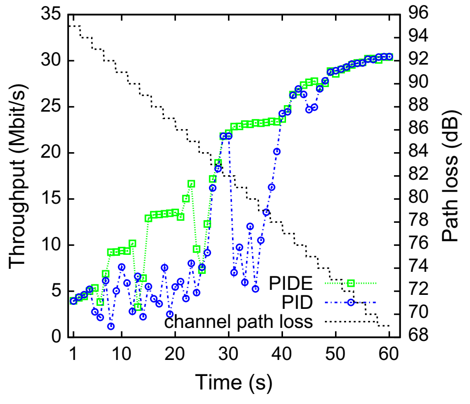
changes of channel conditions happen. Another observation is that in these experiment scenarios PIDE shows better responsiveness than Minstrel, which is shown when comparing Figs. 5 and 12.

4.3.3. Channel quality progressive increase/decrease

Fig. 13 shows the achieved throughput for PID and PIDE as the link quality linearly increases (i.e., decrease in attenuation)/decreases (i.e., increase in attenuation). Although PIDE does not solve the significant throughput drops in both cases, it is still very obvious that PIDE by far performs better than PID in these scenarios. The reason being PIDE selects the optimal rate quicker and stays at the optimal rate most of the time, as shown in the history of rate selection when channel quality decreases (in Fig. 14(b)); in contrast PID has fewer time slots in which it uses the optimal rate, as shown in Fig. 14(a). Another observation is that in terms of the optimal rate usage PIDE and Minstrel are very similar, as shown in Figs. 7(a) and 14(b). Fundamentally, this is because the verification mechanism in PIDE solves the problem of rate hoping in PID.

4.3.4. Over-the-air experiments

To validate results of our conducted experiments, we also revisit the two over-the-air experiment configurations. Fig. 15 shows the performance results in both configurations. As shown in the figures, the results confirm our findings from the conducted experiments, showing that PIDE performs much better than PID and outperforms Minstrel. The absolute throughput difference for the same pair of nodes can be explained by the frequency-selective fading, as described in [16,17]. Fig. 16 (refers to PID and PIDE) shows the number of packets sent and succeeded at each rate (for the semi-controlled experiments), which again confirms that PIDE with the verification mechanism is much better in selecting the optimal rate and staying at this rate. Similar pattern in rate selection is also observed for the *in-the-wild* experiments.


(a) Quality linearly increase

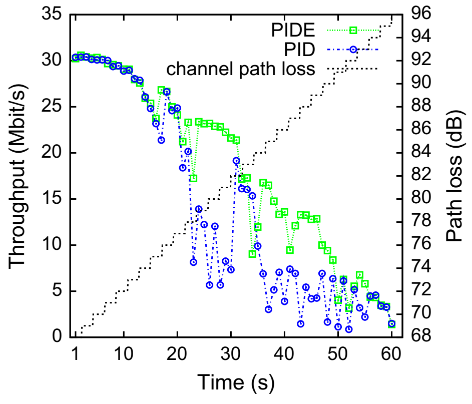
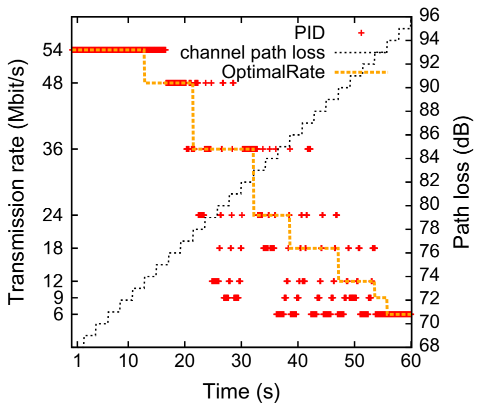

(b) Quality linearly decrease

Fig. 13. Performance as channel linearly changes.


(a) Rate selection for PID.

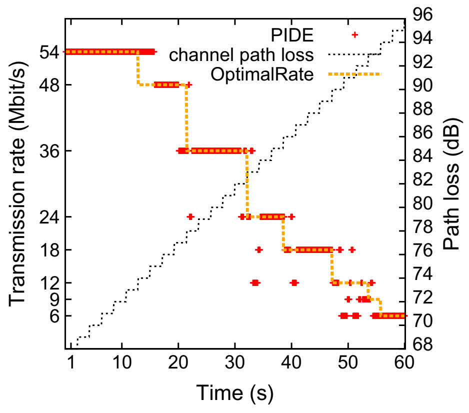
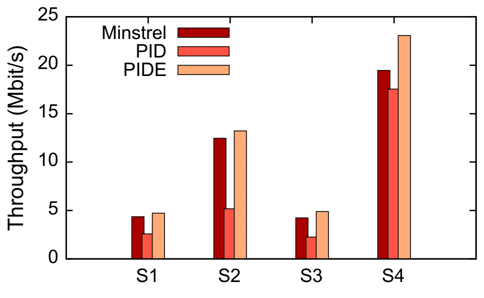

(b) Rate selection for PIDE.

Fig. 14. Rate selection as channel decreases.


(a) Semi-controlled

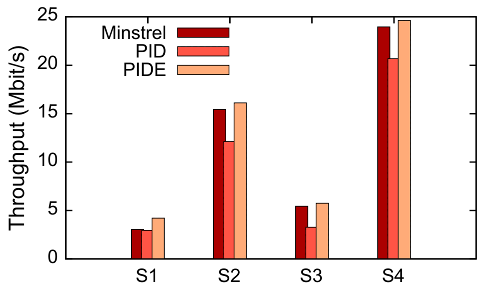

(b) In-the-wild

Fig. 15. Over-the-air performance.

4.4. Discussion

Based on the experience gained from investigating and improving the PID rate control mechanism, we could summarise three lessons learned in terms of using frame loss ratio (FLR) as a metric in designing rate control mechanisms.

First, the mechanism should increase the rate if FLR of the current rate is below a threshold. However it needs to check before switching the rate that the new rate will provide better throughput (or is still below the target FLR, depending on the goal of the network) than the current rate. This is because the current channel conditions may not be able to support such a rate increase. This check

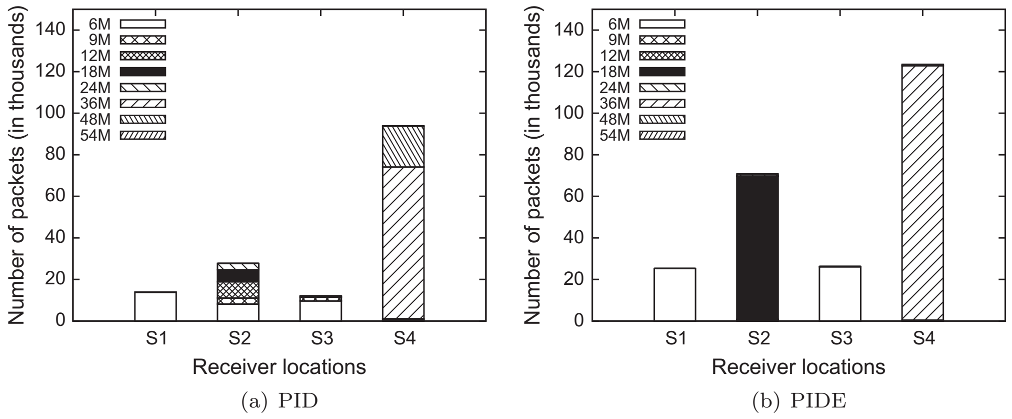


Fig. 16. # of packets succeeded at each rate.

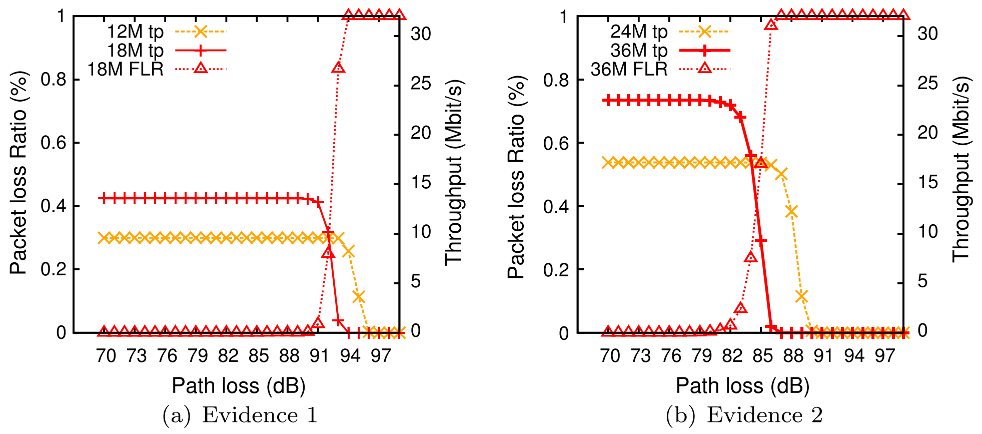


Fig. 17. Relation between throughput and FLR.

could prevent oscillation in rate selections, as we have seen in PID.

Second, rate control mechanisms should decrease the rate if FLR of the current rate exceeds the target threshold (e.g., in PID this target FLR is set to 14%). This is based on the assumption that a rate with such FLR or greater achieves lower throughput than its adjacent lower rate. But this might not be the case in some scenarios. Fig. 17 shows two examples of the relation between throughput and FLR of two adjacent fixed rates at different path loss values; that is, 12 and 18 Mbit/s in Fig. 17(a), and 24 and 36 Mbit/s in Fig. 17(b). These figures show two evidences of how PID using 14% as the target FLR for all rates failed under the above assumption. At the path loss of 92 dB in Fig. 17(a) and 84 dB in Fig. 17(b), we can see that the FLRs at these path loss values are both above 20% for the rate of 18 and 36 Mbit/s, respectively. Therefore, PID will drop the rate under the above assumption. However, if we look closely at the throughput that each respective fixed rate is achieving, we will notice that even with FLR above 20% they still outperform their adjacent lower rates. Hence, we conclude that the above assumption does not hold true in some cases.

Third, assuming a unique FLR threshold for all rates is dangerous. We could see from Fig. 3(a) that the absolute maximum throughput is the envelop of throughputs achieved by all the fixed rates. Hence the optimal FLRs to switch to a lower rate are those crossing points of a higher rate and its next lower rate. When we compared the FLR of these throughput points in Fig. 3(a), we found that they are all different. Therefore, PID using a fixed target FLR of 14% for all rates should fail to achieve maximum throughput. For example, as shown in Fig. 17(a) and (b), to cater for these exceptions, the FLR should be set to 25%, but with this high loss ratio PID will fail to change rate in other scenarios where 14% is a good choice. As we concluded from many evaluation results, there is no single FLR value that would be best for all scenarios. In literature, authors of RRAA [18] also notice that there is no single optimal FLR threshold for all the rates.

5. Collision-aware rate control

Due to the shared nature of the wireless medium, collisions can occur whenever two or more nodes transmit simultaneously. However, this important issue is not addressed by the two rate control mechanisms in the mac80211 framework. In this section, we discuss the issue of collision, propose a solution to address this issue and evaluate how well the proposed solution works.

5.1. Collisions: When, How and Why?

In wireless networks, there are two types of collisions: *collisions due to contention* and *collisions due to hidden terminals*.

5.1.1. Contention collisions

Contention collisions, as the name suggests, occur more often in highly congested networks when a large number of carrier-sensed wireless nodes compete for channel access. As they can carrier-sense each other and the Distributed Coordination Function (DCF) coordinates the medium access, this type of collision only occurs when the backoff time of two or more stations expire at the same time and thus they try to transmit simultaneously, causing collisions.

Fig. 18 shows an example of such collisions, which occurs when the backoff time of two stations expire at the same time even they can carrier-sense the busy medium. In this case, station A selects a backoff time (i.e., Backoff time A) that is the same as the remaining portion of the backoff time for B; therefore they count down to 0 and start the transmission at the same time. As illustrated in this example the collision occurs only when the backoff time of A is equal to the remaining time of the Backoff time of B. The probability of such occurrence in the first transmission is 1/16, as the minimum contention window size of 802.11a network is 16.

With further study we found that there are a few characteristics of this type of collision.

- *The probability of the collision is independent of the selected transmission rates.* As shown in Fig. 18 this type of collisions occur exactly when the backoff time of two or more stations expire at the same time. Therefore, the collision occurs regardless of the transmission rate a station uses.
- *In congested networks the probability of the collision depends on the number of stations that are competing for the channel.* Assuming τ is the probability of an individual station transmitting in a randomly selected slot time causing collision, the overall collision probability [19] of the whole network can be represented as

$$P(\tau, n) = 1 - (1 - \tau)^{n-1} \tag{9}$$

where n is the number of the stations that are competing for channel access. Again the collision probability does not depend on the transmission rate.
- *In the case of contention collisions the frame loss ratio (FLR) of all transmission rates are equal.* If contention collision is the only cause of frame loss, then the collision probability is the same as the FLR, because collided packets are the reason of the increased FLR in this situation. As shown in Eq. (9) the collision probability $P(\tau, n)$ is independent of the transmission rate, and therefore the FLR is also independent of the transmission rate. Hence the FLR of all transmission rates are equal.
- *Collisions due to contention have no impact on rate control mechanisms that use throughput as the metric when comparing the performance of different rates.* Minstrel is one of these mechanisms that adapt transmission rates according to the estimated achievable throughput of each transmission rate. It calculates throughput using the probability of success, P (or $1 - FLR$), and the minimum transmission time for sending a fixed size frame in the perfect channel conditions, $T_{tx\_perfect}$, as discussed in Eq. (1). We argue that in the case of contention collisions all transmission rates have the same FLR. This means throughput depends only on the transmission rates, which are used to compute the $T_{tx\_perfect}$. Generally the higher the transmission rate, the shorter the transmission time required to deliver a frame. These properties stay the same even in case of contention collisions. Therefore, Minstrel prefers the highest transmission rate rather than dropping to lower rates that would increase the frame airtime.
- *Contention collisions can cause problems for rate control mechanisms that use FLR as the metric, if the algorithm is simplistic.* These mechanisms usually predefine one or more target FLR thresholds and probe the performance of different rates to minimise the difference between the target FLR threshold and the actual achieved FLR. Within the mac80211 framework PID is an example of such a mechanism; it sets the target threshold to 14% and adapts the rate to meet the target. However PID employs a rather simple algorithm on rate adaptation, which causes oscillation, as discussed in Section 4.1. Furthermore, using FLR as the metric for rate adaptation is not as straight-forward as throughput, as it is difficult (if not impossible) to determine a unique FLR that will achieve the optimal throughput

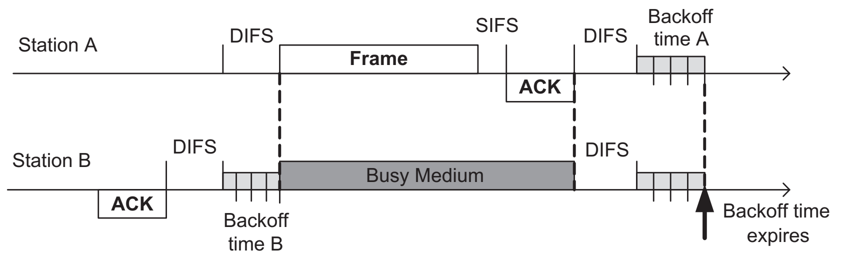


Fig. 18. Example of contention collision.

for different rates; this is an issue we have discussed in Section 4.4. More importantly poor performance will occur for FLR based mechanisms when the number of contention collisions is high. A high number of collisions cause a high FLR which urges FLR based mechanisms to decrease the rate to the base rate. On the contrary, throughput based mechanism do not behave like this under such conditions.

5.1.2. Hidden-terminal collisions

Contention collisions occur only at the beginning of a transmission when the backoff time expires at the same time, whereas collisions due to hidden-terminals can occur at any time during transmissions. This type of collisions happens when two or more nodes cannot carrier-sense each other, but they try to send data to the same receiver causing frames to collide at the receiver.

From the equations defined in the IEEE 802.11 standard, we summarise three characteristics of this type of collisions.

- *The collision probability is inversely proportional to the transmission rate.* In hidden terminal scenarios, collisions can occur at any time during a transmission of another node. If the probability of a hidden node initialising a frame transmission is p, then the probability of collision is $p \cdot t$; where t is the frame transmission time. From Eq. (4), we know that t depends on the bit rate r (as *NDBPS*) used for the transmission. Therefore, t can be represented as $a/r + b$ (a and b are coefficients), and the probability of collision is

$$P_{collision} = p \cdot (a/r + b) \tag{10}$$

  Because a hidden node can initialise a frame transmission at any time during the transmission of another hidden node, the collision probability is not dependent on the p. However, it depends on the selected transmission rate r and it is inversely proportional to this rate. This means frames transmitted at a higher rate are less likely to collide.
- *Under this type of collisions, a lower rate has higher frame loss ratio (FLR).* Let us assume two transmission rates r_1 and r_2 with the corresponding collision probability P_{r_1} and P_{r_2}. If a node has N transmission attempts using rate r_1 with collision probability of P_{r_1}, then the number of failed attempts due to collision are $N \cdot P_{r_1}$ and the frame loss ratio is $(N \cdot P_{r_1})/N$, which is P_{r_1}. Therefore, in the case of hidden-terminal collisions the *FLR* of r_1 is equal to its collision probability; i.e., $FLR_{r_1} = P_{r_1}$. The same is true for rate r_2 and thus we have $FLR_{r_2} = P_{r_2}$. From the last characteristic of hidden-terminal collisions, we know that the collision probability is inversely proportional to the transmission rate. If $r_1 > r_2$, then $P_{r_1} < P_{r_2}$ and therefore $FLR_{r_1} < FLR_{r_2}$. This implies a lower rate has a higher FLR when collisions are only caused by hidden terminals.
- *Mechanisms using FLR as the rate control metric do not work well under this type of collisions.* The hidden-terminal problem will increase the FLR, and a rate control mechanism will drop the rate when the FLR is higher than the predefined threshold. Because a lower rate has higher FLR under this type of collisions, lowering the rate will incur even higher FLR and therefore the base rate will be used. In this case, lower rates use more air time which increases the probability of collisions. Therefore, rate control mechanisms that use FLR as the metric, such as PID, do not work well under hidden-terminal collisions.
- *Mechanisms using throughput as the rate control metric can tolerate hidden-terminal collisions.* On the other hand, mechanisms that select rates based on their achievable throughput will not drop the rate due to the increase of FLR caused by the collisions. This is because a lower rate has a higher FLR and a lower bit rate. The throughput metric calculated based on these two parameters ensures that it will not decrease the rate in the hidden terminal collision environment. Minstrel is an example of such rate control mechanisms.

5.1.3. Discussion

In conclusion, a rate control mechanism should decrease the rate if the channel quality degrades, but it should not decrease the rate because of the high FLR caused by collisions. When comparing Minstrel and PID, we found that PID performs badly in collision scenarios because it decreases the rate to the base rate (due to the increase of FLR).

The problem of contention collisions can be reduced by using an enhanced MAC protocol [20], which gradually decreases the contention window rather than resetting it to the minimum after every successful transmission. We believe this approach can improve the performance of rate control mechanism in congested networks.

In this paper, we focus on the hidden-terminal collisions problem.

5.2. Collision aware mechanism

To address the problem of hidden terminals, we can adaptively activate the RTS/CTS mechanism depending on whether using the mechanism achieves higher throughput. For *normal* data frames, we turn off RTS/CTS for the best rate (i.e., r_0 in MRR), but randomly turn it on with a probability of 10% for all retry rates in the MRR. In a collision free environment retry rates are only used when the best rate fails. Frame loss may be due to hidden terminal collisions. Therefore we turn it on with a small probability to probe the performance statistics for having RTS/CTS turned on; so that we can detect the hidden terminal problem earlier. For *lookaround* frames [7], RTS/CTS is not used for the random rate (higher than the current rate), because it will only be successful when link quality increases. If this random rate fails the frame will be sent with the best rate

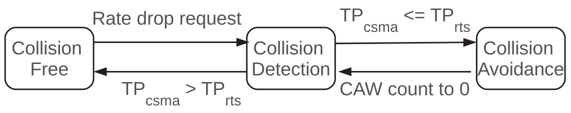


Fig. 19. State machine of the proposed mechanism.

without RTS/CTS. The other retry rates use the RTS/CTS mechanism similarly to normal frames.

Fig. 19 shows a state diagram for the collision-aware mechanism. When the wireless channel is collision free, data frames are sent according to the retry preference defined in the MRR; that is, within a RAP frames retry at a rate (e.g., r_i) for c_i times before attempting other lower rates. After every RAP the throughput statistics of all supported rates are calculated (for both with or without RTS/CTS) to work out the best performed rate for the next RAP. For each frame, the RTS/CTS is turned on or off for each rate in the MRR chain, but in a rate adaptation period, many frames are transmitted with different MRR settings. Retry rates turn on the RTS/CTS with a probability as discussed above. Therefore from a rate perspective, four pieces of information are gathered including attempts with RTS/CTS on, successes with RTS/CTS on, attempts with RTS/CTS off, successes with RTS/CTS off. Based on this information, for both with or without RTS/CTS, throughput is calculated. When a lower rate achieves higher throughput than the first rate (i.e., r_0), a rate drop request is issued. The throughput of the current rate for both with and without RTS/CTS is compared, and when the throughput for having RTS/CTS is greater than without it (i.e., $TP_{rts} \geqslant TP_{csma}$) we then conclude that collision is occurring and the RTS/CTS mechanism is turned on in the next RAP rather than dropping the rate. The collision avoidance window (CAW) specifies for how long the RTS/CTS mechanism should be used and it defaults to one RAP initially and grows exponentially if collisions still exist when CAW expires. The maximum CAW is 16 considering that its large value could cause poor performance when the collision does not last long. As long as the CAW is bigger than zero, the mechanism is in the collision avoidance state. When CAW reaches zero, the mechanism enters the collision detection state and the RTS/CTS will be turned off for one RAP. At the end of the RAP, the mechanism compares TP_{rts} and TP_{csma}. In the case when TP_{rts} is smaller than TP_{csma} there is no collision and rate adaptation follows the normal retry preference as described earlier.

It is obvious that using the RTS/CTS mechanism introduces an overhead to the rate control mechanism. However we use achievable throughput of all rates as the metric for evaluating their performance. The throughput gain and the overhead of the RTS/CTS exchange are considered when deciding on a rate. The computation of TP_{rts} and TP_{csma} is based on Eq. (7). Specifically the transmission time of RTS and CTS frames is considered when RTS/CTS is utilised.

We implemented the collision aware mechanism in the Minstrel and PIDE rate control mechanisms on the mac80211 framework. As PID suffers from obvious oscillation in rate selection, we argue that adding the collision aware mechanism to PIDE will be more meaningful for the comparison.

5.3. Experiments and discussions

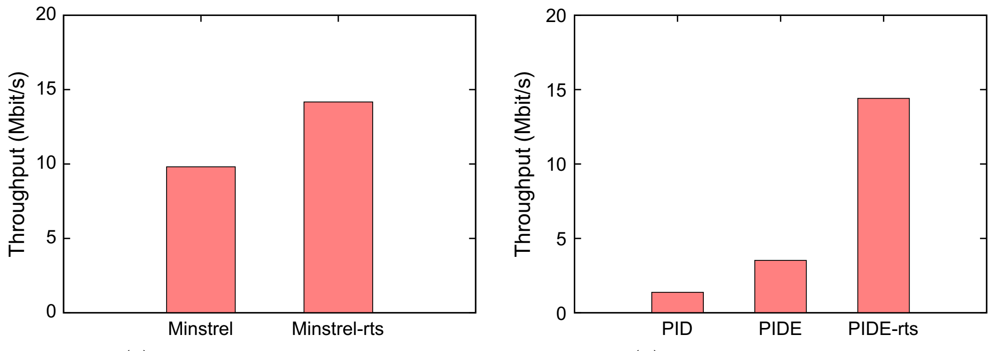
In this section, we describe two experiments carried out to evaluate the performance of the proposed collision aware mechanism in the hidden-terminal environment.

5.3.1. Hidden-terminals with equal link quality

For the first experiment, we put together a hidden-terminal scenario ($Sender \rightarrow Receiver \leftarrow Interferer$) using our conducted testbed, as discussed in Section 3. To minimise the impact of capture effect, we set the same link quality for both links.

Fig. 20 shows a performance comparison of all mentioned rate control mechanisms in the hidden-terminal scenario. We can clearly see the performance gains achieved by adaptively activating RTS/CTS. Minstrel with the proposed mechanism (i.e., Minstrel-rts) achieves 47% higher throughput than without it. For PIDE, the improvement is even higher, almost ten times. In addition, PIDE outperforms the PID under collisions as we expected. But we can clearly see that Minstrel performs better than PIDE if collisions occur. This clearly indicates that the functionality of PIDE is limited in addressing the rate oscillation problem and the collision aware mechanism is an essential component to further enhance its performance. Both Minstrel-rts and PIDE-rts use the same collision aware mechanism, hence achieve similar throughput. We will explain later that the throughput achieved by collision aware mechanisms is very close to the optimal.

One of the surprising observations from Fig. 20 is Minstrel's performance compared to PID and PIDE under collision scenarios. This performance difference is



(a) Minstrel and Minstrel-rts. (b) PID, PIDE and PIDE-rts.

Fig. 20. Performance in hidden-terminals scenario.

produced by how they react to collisions. Even under collisions, Minstrel still tries to estimate each rate's achievable throughput and selects the one that provides the highest throughput. Whereas PID and PIDE select rate purely based on FLR, which makes them vulnerable to collisions (as discussed later). Based on further investigations of the statistics produced by Minstrel, we find that even with about 50% of FLR, Minstrel still does not decrease the rate. This observation confirms our previous analysis. By adaptively activating RTS/CTS, we bring down the frame loss due to collisions to approximately 3%. This coincides with the way the mechanism works. That is, for every 16 CAWs using RTS (zero loss due to collisions), the mechanism turns off RTS for one CAW in which 50% loss is generated. Therefore, this loss ratio is averaged out across the whole one second of statistics ($50\%/17 \approx 3\%$). This indicates that the collision aware mechanism has already achieved 97% of the optimal throughput. Multiple experiments confirm this analysis. It should be noted that using RTS/CTS we can achieve the optimal throughput in hidden-terminal scenarios, without frame losses. However, it will result in significant overhead when collisions occur only occasionally. It is possible to further reduce this loss ratio by increasing the length of CAW. However, increasing the length of the CAW means that we expect the collisions to last for longer time and more transmissions are sent with RTS/CTS. Given the fact that collisions are probabilistic events and there is no knowledge about their occurence, we limit the maximum CAW length to 16. It is a tradeoff parameter, which we plan to investigate further in the future.

The poor performance of PID is partially caused by the oscillation problem in the rate selection, as discussed in Section 4. The direct effect of collisions is the increase of FLR. When the FLR is greater than 14%, PID drops the rate, which makes collisions even worse because a lower rate has longer transmission time (therefore higher FLR). A higher FLR will force PID to further decrease the rate, until the base rate is reached. The traces of the dataset confirm that PID is using the base rate for the majority of the transmissions, therefore achieving such low throughput. Adding a mechanism in PIDE to check whether the proposed rate achieves better throughput does address the rate oscillation problem in collision scenarios. This mechanism only uses probing frames to check the performance of the proposed rate. Collision is a probabilistic event, therefore there might be a slight chance that a lower rate without collisions outperforms a high rate with collisions. Hence, PIDE will slowly drop the rate eventually under frequent collisions. This is the reason why Minstrel outperforms PIDE in the collision scenario. It is obvious that if we can adaptively activate RTS/CTS as proposed, then we can efficiently avoid collisions and achieve significant throughput gain.

5.3.2. Impact of interference load

The last experiment is carried out by sending saturated traffic from both the sender and interferer. Next, we vary the interferer's traffic load to study the impact of varying interfering load on the performance of the mechanism. We use the same topology setup as in the previous experiment. The sender will try to saturate the wireless link, but we vary the offered load on the interferer. We set the interferer to use a fixed rate during the experiment to avoid the impact of rate control on the interferer. Due to the use of RTS/CTS in collision scenarios, we expect both Minstrel and PID with the proposed mechanism will outperform their corresponding counterparts.

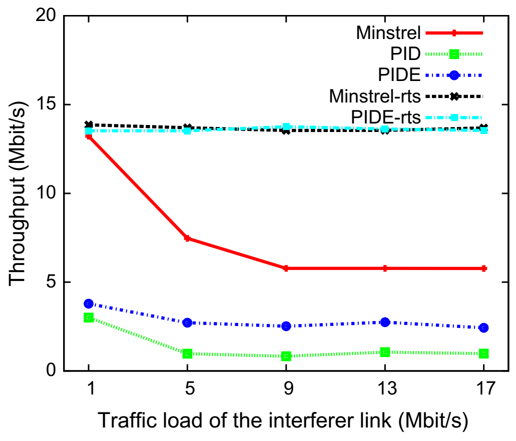


Fig. 21. Impact of different interference loads.

Fig. 21 confirms the reasoning. The results generally agree with the findings from the previous experiments, with one exception. That is, PID and PIDE suffer significant problem even with low load of interference, whereas Minstrel seems to have the ability to tolerate low load of interference and the performance degrades if the interference load increases.

This observation can be explained with the way these mechanisms adapt transmission rates. PID and PIDE both react directly to the changes of FLR, only difference is in how fast they react. In contrast, Minstrel selects the rate based on the estimated achievable throughput of each rate. The impact of sudden changes in FLR due to collisions does affect the performance of Minstrel. The more robust metric used in Minstrel is not enough to address the collision problem if the interference load significantly increases. This set of experiments again suggests that adaptively engaging the RTS/CTS mechanism is a better way to achieve high throughput in collision scenarios.

5.4. Limitations

There are four limitations regarding the collision aware mechanism.

(i) The overhead of turning on RTS/CTS for one CAW if the rate drop request is caused by normal decrease of channel quality. This overhead is relatively small, since we only need to probe the performance with RTS/CTS when there is a rate drop request and the duration is only for one CAW (100 ms). Through this probing window, the mechanism can gather the statistics to compare whether engaging RTS/CTS can bring throughput gain, not sole overhead. RTS/CTS exchanges will only be activated if necessary.

(ii) The overhead of using RTS/CTS when collisions go away. We double the CAW every time collisions are detected at the end of the next RAP after CAW expires. Essentially, this means the collisions exist longer than this CAW. We extend the duration for the use of RTS/CTS exchanges if collisions persist. However, the standard RTS/CTS overhead is generated if collisions exist for a shorter time than the extended CAW. To address this problem, we limit the maximum CAW doubling to 16. We plan to study the impact of this parameter in the future work.

(iii) The throughput drop caused by turning off RTS/CTS for one RAP to probe the performance without using RTS/CTS exchanges. In this case, if the collisions still exist, then there will be significant throughput drop for one RAP. This is a trade-off mechanism to ensure we can quickly react to the case when collisions disappear. Given the fact that the characteristics of collisions in over-the-air environment is not continuous over a long duration, we argue that this trade-off mechanism is necessary and sufficient.

(iv) The collision due to contention is qualitatively discussed in the paper, but quantitative evaluation has not been conducted due to two reasons: To begin with, to create a testbed that produces contention collisions is extremely difficult and requires a significantly large number of nodes. Secondly, the control of contention collision is difficult, if it is not impossible; as aforementioned, a contention collision only occurs when backoff time of two or more nodes expire at the same time and they start to access the channel. We do not have access to a testbed that allows us to have a fine control of contention collision. Therefore, we leave this type of evaluations as future work. Alternatively, we have attempted to perform such experiments in a simulation environment, such as ns3. However, precise information about transmission status is not supported in the simulation environment. Thus we have difficulty to port the existing implementation of Minstrel on ns3, and even we did port it the information about failure attempts of each rate would be missing. This information is important for the multi-rate retry mechanism. Without this information, the throughput metric calculation for each rate in Minstrel is not accurate, which impacts on rate adaptation. Therefore, we are not in favour of studying this type of collisions in simulation environments.

6. Related Work

The MAC layer rate control mechanism has been an active area of research for many years. A number of mechanisms have been proposed to adaptively select transmission rates according to the channel conditions. Based on the type of *metric* used, rate control algorithms can be classified into three groups: ACK (Acknowledgement), SNR (Signal-to-Noise Ratio) and BER (Bit Error Rate) based mechanisms.

ACK based mechanisms, including RRAA [18], AMRR [5], SampleRate [6], Onoe [4], Minstrel [7], PID [11] and a rate control mechanism in [21], adapt rates according to the results of transmissions (e.g., frame loss ratio and average transmission time), which are estimated based on received acknowledgements. ACK based mechanisms are low cost, because the metrics can be easily calculated by monitoring ACKs. SampleRate [6] is based on the estimated transmission time required to successfully deliver a frame. Minstrel on the other hand calculates the estimated maximum achievable throughput based on the effective bandwidth and FLR of each rate. RRAA and PID directly use the FLR to make rate adaptation, i.e. they increase the rate when FLR is below a threshold and decrease the rate when FLR exceeds another threshold. Similarly, AMRR and Onoe decrease the rate when FLR is greater than a threshold, however, they increase the rate more carefully in order to avoid the rate oscillation problem. For example, when the FLR is less than a threshold, they only increase a credit. After the credit exceeds another threshold, they will upgrade the transmission rate. However, this approach is conservative in some scenarios. For example, they need multiple rate adaptation periods (RAPs) to upgrade to the next higher rate, which makes it not responsive to events like fast fading. To address the rate oscillation problem in PID, the proposed verification mechanism in PIDE probes the performance of the proposed lower rate in one RAP window before droping the rate.

SNR based mechanisms (or SINR, Signal-to-Interference-and-Noise Ratio, if we consider the impact of interference) measure the signal-to-noise ratio to determine an appropriate transmission rate. These mechanisms include RBAR [22], CHARM [23] and FARA [16]. BER based mechanisms use a more fine-grained metric, which uses the average number of bit errors in each frame. An example of such a mechanism is SoftRate [24]. Both SNR and BER mechanisms require accurate measurement of the value of SNR (or SINR) and BER. It is arguably difficult to estimate the SNR without receiver's feedbacks and to measure the BER using off-the-shelf radio cards [25]. In CHARM, the authors proposed an approach for the sender to estimate SINR without needing receiver's feedback. However, they mentioned there are a number of factors, which have potential impact on the accuracy of their SINR estimation. Without disclosing further details of their approach, we are not sure how sender estimates interference without asking the receiver. Finally, their results show that CHARM only achieves a minor marginal throughput gain comparing to SampleRate or AMRR. In [3,8], we show that Minstrel outperforms SampleRate and AMRR significantly and achieves a performance close to the optimal rates. This implies ACK mechanisms can also achieve better performance than SNR or BER approaches.

In the case of performing rate control under hidden-terminal collisions, existing work tends to rely on using RTS/CTS protocol. CARA [26] and A-RTS [18] adaptively activate RTS/CTS exchange. However, the approaches are based on different heuristics and these heuristics cannot guarantee that turning on RTS/CTS can really bring throughput gains. The reason is that they do not compare the throughput loss introduced by the overhead of RTS/CTS to its throughput gain. In addition, they are per-frame rate control mechanisms (designed for low latency systems, therefore require specific hardware) and are not suitable to be implemented in Linux.

7. Conclusion

The mac80211 framework has been adopted in millions of computers running Linux. It provides fundamental frame management and information sharing between different layers. Unfortunately, a comprehensive analysis of this framework and its mechanisms for MAC-layer rate control does not exist. This paper provides a comprehensive study of this framework and the two rate control mechanisms: Minstrel and PID. We showed that PID has poor performance compared to Minstrel; we then proposed and evaluated a PID enhancement (PIDE) that improves the performance. Apart from that we presented a collision-aware rate control mechanism, which is able to detect hidden terminal collisions, and dynamically engage the RTS/CTS mechanism to prevent frame losses due to collisions. The proposed solution dynamically turns on/off RTS/CTS after probing the throughput performance in each scenario. The evaluation shows that it can enhance the Minstrel's performance by more than 47% and PID by 10 times. Finally we discussed a lesson learned from evaluating and improving rate control mechanisms, i.e., frame loss ratio is not the best metric for rate adaptation.

Acknowledgments

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program; and the Queensland Government, Australia.

References

[1] Cisco, Inc., Cisco visual networking index: Global mobile data traffic forecast update 2014–2019 white paper, 2015.
[2] IEEE Computer Society, IEEE Standard 802.11-2012, 2012.
[3] W. Yin, K. Bialkowski, J. Indulska, P. Hu, Evaluation of MadWifi MAC layer rate control mechanisms, in: Proceedings of IWQoS '10, Beijing, China, 2010.
[4] Onoe Specification, Madwifi Project.
[5] M. Lacage, M.H. Manshaei, T. Turletti, IEEE 802.11 rate adaptation: a practical approach, in: Proceedings of MSWiM'04, ACM, Venice, Italy, 2004, pp. 126–134.
[6] J.C. Bicket, Bit-rate Selection in Wireless Networks, MIT Master Thesis, 2005.
[7] LinuxWireless Project, Minstrel Specification. <http://wireless.kernel.org/en/developers/Documentation/mac80211/RateControl/minstrel>.
[8] W. Yin, P. Hu, J. Indulska, K. Bialkowski, Performance of mac80211 rate control mechanisms, in: Proceedings of MSWiM '11, Miami, FL, USA, 2011.
[9] W. Yin, P. Hu, J. Indulska, M. Portmann, J. Guerin, Robust MAC-layer rate control mechanism for 802.11 wireless networks, in: Proceedings of the 2012 IEEE 37th Conference on Local Computer Networks (LCN 2012), LCN '12, 2012, pp. 419–427.
[10] P. Salvador, S. Paris, C. Pisa, P. Patras, Y. Grunenberger, X. Perez-Costa, J. Gozdecki, A modular, flexible and virtualizable framework for IEEE 802.11, in: Future Network Mobile Summit (FutureNetw), 2012, pp. 1–8.
[11] LinuxWireless Project, PID Specification. <http://wireless.kernel.org/en/developers/Documentation/mac80211/RateControl/PID>.
[12] W. Yin, P. Hu, J. Indulska, K. Bialkowski, A method to improve adaptability of the Minstrel MAC rate control algorithm, in: Proceedings of UIC '10, Xi'an, China, 2010.
[13] B. Sklar, Rayleigh fading channels in mobile digital communication systems. I. Characterization, Commun. Mag., IEEE 35 (7) (1997) 90–100.
[14] J. Camp, E. Knightly, Modulation rate adaptation in urban and vehicular environments: cross-layer implementation and experimental evaluation, in: Proceedings of MobiCom '08, ACM, San Francisco, California, USA, 2008, pp. 315–326.
[15] X. Ma, L. Yang, G. Giannakis, Optimal training for mimo frequency-selective fading channels, IEEE Trans. Wireless Commun. 4 (2) (2005) 453–466.
[16] H. Rahul, F. Edalat, D. Katabi, C.G. Sodini, Frequency-aware rate adaptation and MAC protocols, in: Proceedings of MobiCom '09, ACM, Beijing, China, 2009, pp. 193–204.
[17] D. Halperin, W. Hu, A. Sheth, D. Wetherall, Predictable 802.11 packet delivery from wireless channel measurements, in: Proceedings of SIGCOMM '10, New Delhi, India, 2010, pp. 159–170.
[18] S.H.Y. Wong, H. Yang, S. Lu, V. Bharghavan, Robust rate adaptation for 802.11 wireless networks, in: Proceedings of MobiCom '06, ACM, Los Angeles, California, USA, 2006, pp. 146–157.
[19] G. Bianchi, Performance analysis of the IEEE 802.11 distributed coordination function, IEEE J. Sel. Areas Commun. 18 (3) (2000) 535–547.
[20] I. Aad, Q. Ni, C. Barakat, T. Turletti, Enhancing IEEE 802.11 MAC in congested environments, Comput. Commun. 28 (14) (2005) 1605–1617.
[21] R. Aggarwal, P. Schniter, C.E. Koksal, Rate adaptation via link-layer feedback for goodput maximization over a time-varying channel, IEEE Trans. Wireless Commun. (2009).
[22] G. Holland, N. Vaidya, P. Bahl, A rate-adaptive mac protocol for multi-hop wireless networks, in: Proceedings of MobiCom '01, 2001.
[23] G. Judd, X. Wang, P. Steenkiste, Low-overhead channel-aware rate adaptation, in: Proceedings of MobiCom '07, ACM, Montreal, Quebec, Canada, 2007, pp. 354–357.
[24] M. Vutukuru, H. Balakrishnan, K. Jamieson, Cross-layer wireless bit rate adaptation, in: Proceedings of SIGCOMM '09, ACM, Barcelona, Spain, 2009, pp. 3–14.
[25] E. Ancillotti, R. Bruno, M. Conti, Design and performance evaluation of throughput-aware rate adaptation protocols for IEEE 802.11 wireless networks, J. Perform. Eval. 66 (2009) 811–825.
[26] J. Kim, S. Kim, S. Choi, D. Qiao, CARA: collision-aware rate adaptation for IEEE 802.11 WLANs, in: Proceedings of INFOCOM '06, Barcelona, Spain, 2006, pp. 1–11.

Wei Yin is an engineer at Air Force Equipment Research Institute, China. He received his Ph.D. from the University of Queensland in 2012. His research areas include wireless networks and BGP security.

Peizhao Hu is an assistant professor in the Department of Computer Science at Rochester Institute of Technology, New York, USA. His research focuses on Mobile and Pervasive Computing, wireless mesh networking. Dr. Hu had served as technical program committee and organizing committee for a number of conference and workshops, such as PerCom, LCN, MDM, AINA, UIC. In addition, he had served as technical committee or reviewer for international journals, including Journal of Pervasive and Mobile Computing (Elsevier), Journal of Computer Communications (Elsevier), Multimedia Systems Journal (ACM), Mobile Networks and Applications (Springer). He has authored more than twenty journal and conference papers in his research expertise.

Jadwiga Indulska is a Professor of Pervasive Computing in the School of Information Technology and Electrical Engineering at the University of Queensland, Brisbane, Australia. Her research interests are in the areas of computer networks, distributed computing, and pervasive computing. Over the last 15 years, her research has addressed many problems in pervasive and autonomic computing, including context information models for context-aware applications, autonomic management of context information, privacy of context information, software engineering of context-aware applications, balancing user control and software autonomy, and autonomic, rapidly deployable mesh networks. She has led research projects on interoperability of distributed applications, mobile computing, pervasive computing, and autonomic networks at the DSTC, an Australian Government funded Collaborative Research Centre on Distributed Systems Technology (1992–2005), and at NICTA (National Centre of Excellence in Information and Communication Technology). She has served on editorial boards of journals on pervasive computing including the Elsevier Journal of Pervasive and Mobile Computing and on numerous conference program committees. She has been a very active member of the PerCom organizing committee (IEEE International Conference on Pervasive Computing and Communications).