

# Automatic IEEE 802.11 Rate Control for Streaming Applications

Ivaylo Haratcherev, Jacco Taal, Koen Langendoen, Reginald Lagendijk, Henk Sips

Faculty of Electrical Engineering, Mathematics, and Computer Science, Delft University of Technology

Mekelweg 4, 2628 CD Delft, The Netherlands

{I.Haratcherev, J.R.Taal, K.G.Langendoen, R.L.Lagendijk, H.J.Sips}@ewi.tudelft.nl

## Abstract

Streaming multimedia content in real-time over a wireless link is a challenging task because of the rapid fluctuations in link conditions that can occur due to movement, interference, and so on. The popular IEEE 802.11 standard includes low-level tuning parameters like the transmission rate. Standard device drivers for today's wireless products are based on gathering statistics, and consequently, adapt rather slowly to changes in conditions. To meet the strict latency requirements of streaming applications, we designed and implemented an advanced hybrid control algorithm that uses signal-strength (SNR) information to achieve fast responses. Since SNR readings are quite noisy we do not use that information to directly control the rate setting, but rather as a safeguard limiting the range of feasible settings to choose from. We report on real-time experiments involving two laptops equipped with IEEE 802.11a wireless interface cards. The results show that using SNR information greatly enhances responsiveness in comparison to statistics-based rate controllers. Finally, we will present the results of an experiment with realtime video streaming to a moving laptop in an office-like environment. Our hybrid control algorithm effectively prevented many packets losses, thereby achieving a much higher video quality than

the statistics based algorithm.

**Keywords:** rate control, MAC layer, SNR, link adaptation, video streaming

## 1 Introduction

It is anticipated that multimedia streaming over the Internet will have a significant share in tomorrow's communications. Also, end users increasingly seek mobility, thus paving the way for extensive deployment of wireless technologies like IEEE 802.11. The joint effect is that support is needed for multimedia streaming over connections that include both fixed and wireless links. In this paper, we focus on the weakest part of such connections: streaming over a wireless link (the last hop). Such a link is the bottleneck for two reasons: First, communication over a wireless channel is simply not able to achieve the same quality (throughput, error rate, etc.) as its wired counterpart, which reduces the quality of the multimedia content that can be delivered. Second, in a mobile environment, the channel conditions can change rapidly due to changing distance between the stations (user mobility), Rayleigh fading, interference and so on. Since multimedia streaming applications must deliver their content in real time, they are very sensitive to jitter in packet delivery caused by retransmissions in the underlying transport protocols. Consequently, when using streaming applications, users experience reduced range compared to the case when less demanding applications like file downloading and web browsing are used.

With today's 802.11 products, the fundamental problems of wireless communication are aggravated by poor handling of the limited and imperfect resources (scarce spectrum, noisy medium) available to the radio. In particular, current transport protocols and device drivers do not actively control the user-available parameters of the 802.11 MAC layer; they use some default values instead. In this paper, we demonstrate that much can be gained by tuning the MAC parameters to the (fluctuating) channel conditions.

The remainder of the paper is organized as follows. The basics of the link adaptation are discussed in the next section. Section 3 gives a description of the existing rate control algorithms. Our improved solution is introduced in Section 4. The experimental results are discussed in Sections 5 and 6. The conclusions and

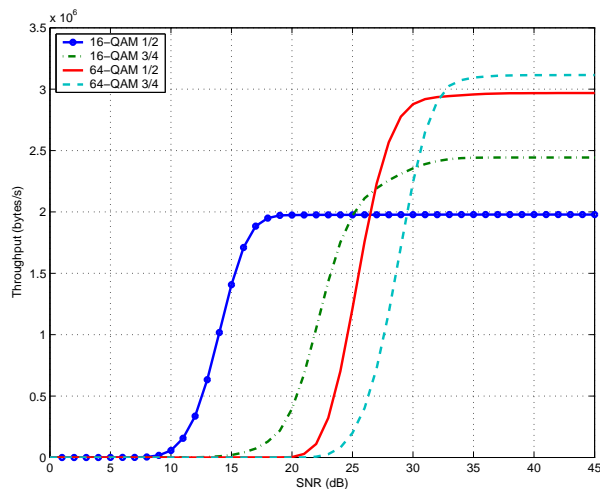


Figure 1: Throughput vs. SNR for some 802.11a modulation schemes.

future plans are presented in Section 7.

## 2 Link adaptation basics

The IEEE 802.11 standard defines several MAC-level parameters (settings) that are available for tuning at the side of the Wireless Network Interface Card (NIC). The most important parameter available for adjustment is the transmit rate. In 802.11a, for example, it can be set to 6, 9, 12, 18, 24, 36, 48 and 54 Mb/s. Each rate corresponds to a different modulation scheme with its own trade-off between data throughput and distance between the stations. This can be seen in Figure 1 - for clarity only the last four modulation schemes are shown. It shows the performance in terms of the throughput for each modulation scheme available in IEEE 802.11a versus the signal-to-noise ratio (SNR). Note that distance is related to SNR as  $SNR \sim \frac{1}{dist^\alpha}$ . More complex modulation schemes like 64-QAM 3/4 offer a larger throughput, but also have increased sensitivity to channel noise, and thus provide a shorter operating range. Usually, one wants to extend the operating range as much as possible and, at the same time, to maximize the throughput. This can be done by proper (automatic) selection of the rate (modulation scheme) that gives the maximum throughput for

certain conditions, for example, by selecting 64-QAM 1/2 at an SNR of 30 dB (Figure 1).

In order to decide which rate is optimal at each specific moment, a control algorithm needs information about the current link conditions, or so-called channel state information (CSI). Since it is difficult to get CSI directly, most rate-control algorithms use some form of statistics-based feedback, for example, user-level throughput (see Section 3). The main disadvantage of this indirect feedback is that it is inherently slow, causing communication drop-outs when the link conditions degrade rapidly (e.g., when the user moves fast). The short-term drop-outs are normally handled by frame retransmissions. This is acceptable for download applications whose main requirement is gross data throughput. The retransmissions, however, lead to a significant increase in (average) packet delay, and the variations in the number of retransmissions cause an increase in the jitter of the packet delay. Streaming applications are very sensitive to long packet delays and high jitter, and less sensitive to the overall throughput of the link (provided of course that this throughput is larger than the minimum throughput that the application requires). Consequently, streaming applications perform poorly under standard automatic rate control. In [4], for example, it is reported that switching from automatic rate control to a manually selected fixed rate extends the maximum distance between stations up to 40% for the flawless display of a video stream over an 802.11a wireless connection.

A logical way to cope with the slow accommodation characteristics of statistics-based feedback methods is to look for methods that use faster feedback, i.e., feedback that quickly provides up-to-date information about the channel status. Such a feedback - the SNR - has been theoretically discussed in previous works, but it has never been used in actual implementations. The main reason for this is that, in practice, it turns out to be very difficult to obtain a reliable estimate of the SNR (see Section 3). In this paper, we discuss an advanced hybrid approach to mitigate the SNR-related problems, and report on a practical implementation of this approach in a novel automatic rate-selection algorithm for IEEE 802.11a wireless connections.

It is important to note that all the CSI-based rate control schemes have one very important disadvantage in common. In case of low or no traffic, the CSI known to the system becomes outdated, therefore disrupting the work of the rate control algorithm. There are two ways to cope with this problem. The first is to use safety mechanisms, like timers to invalidate the CSI after not being updated for certain period of time.

When the CSI known to the algorithm has been marked invalid, the rate controller can either revert to some safe rate setting to send the next packet, or just stay on the last setting used. The other approach to handle the effects of stall traffic, is to create additional small background traffic, so that CSI can be updated regularly. Discussion on the pros and cons of each method would be too extensive and goes out of the scope of this paper. In addition, we are focusing on streaming applications, which provide steady data stream, so in normal operation the CSI never becomes outdated. Our algorithm makes use of the first approach, i.e. it uses timeouts and other safety mechanisms.

### 3 Types of CSI-based rate control

The IEEE 802.11 standard [10] and its supplements do not specify an algorithm for automatic rate selection. So those are the companies manufacturing 802.11 interfaces, that have to come up with the algorithm: they have the freedom to design and implement their own proprietary schemes for control. To our knowledge, all of the known vendors of 802.11 equipment use statistics-based approaches for rate control. The control algorithms can be implemented in software as part of the device driver for the wireless NIC, but also as part of the chipset, which allows for precise control of individual frame (re-)transmissions. In the research community, another class of rate control algorithms has been studied. These control algorithms use SNR-related information as a feedback to improve the sensitivity to changes in link conditions. Up till now, however, such algorithms were not implemented in practical systems, and only simulation results were reported.

In the discussion below we will describe the main representatives of both classes (statistics- and SNR-based) of rate control algorithms. It should be noted that the performance of all algorithms can be improved by differentiating to packet length, because the probability of a packet being corrupted depends on the length of the transmission. Consequently, long packets should be transmitted at a lower rate than short packets.

## 3.1 Statistics-based automatic rate control

An easy way to obtain the necessary information on the link conditions is to maintain statistics about the transmitted data like the frame error rate (FER), acknowledged transmissions, and the achieved throughput. Since these statistics are directly related to the effective user-level data throughput, they inherently guarantee that this throughput is maximized on the long-term. These factors (simplicity and stability) explain the dominance of statistics-based feedback in current 802.11 products. Three basic types of statistics-based rate control can be distinguished: throughput-based, FER-based, and retry-based rate control. The throughput-based approach is the one that uses the most global type of statistic and is the slowest method. The retry-based control uses the most local statistic (number of retries per frame), and is the fastest method. Each statistics-based rate control type is briefly discussed in the rest of this subsection.

### 3.1.1 Throughput-based rate control

In this approach, a constant small fraction (10%) of the data is sent at the two adjacent rates to the current one (an adjacent rate is the next higher or lower one available). Then, at the end of a specified decision window, the performance of all three rates is determined by dividing the number of bytes transmitted at each rate by their cumulative transmission times. Finally, a switch is made to the rate that provided the highest throughput during the decision window. Atheros uses this algorithm in the NIC driver that they provide for their 802.11a products based on the AR5000 chipset.

To collect meaningful statistics, the decision window has to be quite large (i.e., about one second). On the one hand this makes the algorithm resilient to short-lived changes in the link quality caused by, for example, Rayleigh fading. On the other hand, it prevents swift reactions to long-lived changes in link conditions, which noticeably affects the real-time performance of streaming applications.

### 3.1.2 FER-based rate control

In this approach, the Frame Error Rate (FER) of the data stream transmitted over the link is used to select an appropriate rate. The FER can easily be determined since under 802.11, all successfully received data

frames are explicitly acknowledged by sending an ACK frame to the sender; hence, a missing ACK is a strong indication of a lost data frame. By counting the number of received ACK frames and the number of transmitted data frames during a rather short time window, the FER can be computed as the ratio of the two.

The FER can be used to select the rate setting for the next time window as follows [2]:

**downscaling** If the FER exceeds some threshold and the current rate is not the minimal rate, then switch to the next lower rate.

**upscaling** If the FER is close to zero (i.e., below a second threshold), probe the link at the adjacent higher rate with a few (usually even only 1) frames. If all of them get acknowledged, switch to that rate. To prevent the control algorithm from oscillating between two adjacent rates, the upscale action may be prohibited for some time after a downscale decision.

The width of the time window and the thresholds mentioned above are critical for the performance of the FER-based algorithm. The optimal settings of the parameters are dependent on the link and the application, but are generally fixed at design time. Again, this hampers the performance of streaming applications, since a time window tuned for quick responses of typical download applications (to changes in link conditions) yields unreliable FER statistics at low traffic rates. Hence, many frames are transmitted at a non-optimal rate.

### 3.1.3 Retry-based rate control

An improvement over the FER-based approach is to downscale immediately when the MAC is struggling to transmit a frame correctly over the link. That is, to select the next lower rate after a small number of unsuccessful retransmissions (usually 5-10 retries) [8, 14]. This approach is implemented in hardware, as precise control of the rate setting in between retransmissions (of the same frame) is required.

The advantage of the retry-based approach is that it combines a very short response time (a few frames) for handling deteriorating link conditions (downscaling) with a low sensitivity to traffic rates. The price to

be paid is that the control algorithm is rather pessimistic. Relatively short error bursts cause long drops in throughput because upscaling to higher rates takes much longer than downscaling due to the need to collect a meaningful FER and to prevent oscillation.

One other important disadvantage of the retry-based approach is that in case of collisions (when other stations are trying to transmit simultaneously), the algorithm will fall back because of the increase of the retries per frame. First, this will cause an undesired drop in throughput (because we switch to a lower rate), which in fact will add on the loss of the throughput already caused by the contention for the medium. Second, unnecessary fallbacks to low rates cause unfairness to the other users as well, because the additional air-time reduces their throughput (see [5]). Unfortunately, without using additional CSI feedback, there is no way that the control algorithm can distinguish between different causes for a frame being retransmitted (bad link or collisions). Therefore it can not avoid unnecessary switching to lower rates in case of medium contention.

### 3.2 SNR-based automatic rate control

A fundamental limit of indirect, statistics-based feedback is that it classifies link conditions as either "good" or "bad". This binary information provides some notion about the direction in which to adapt the rate setting, but does not suffice to select the appropriate rate at once. This leads to a slow step-by-step accommodation to large changes in conditions, and introduces the risk of oscillation in stable conditions. A better approach is to use direct measurements of the link conditions.

The Signal-to-Noise Ratio (SNR) is directly related to the bit-error rate in the link and, hence, to the FER. Consequently, the SNR is linked to the packet delay and jitter, and the throughput, and holds the potential of providing rich feedback for automatic rate control [1]. Knowing the current SNR and the throughput-vs-SNR curves for each rate setting (e.g., Figure 1) solves the rate-selection problem instantly: we simply switch to the rate with the highest throughput for the current SNR. This can be implemented efficiently by means of a lookup table.

Despite the advantages, SNR-based rate control has not been applied in practice so far. This is mainly

because of three reasons. First, in reality, for certain link conditions the relation between the optimal rate and SNR is highly variable. This is due to the imperfectness of the models describing the radio channel, and also because the link quality depends to some extent on other parameters as well. Second, it is not trivial to obtain a reliable estimate of the SNR of a link. Many radio interfaces provide only an uncalibrated Signal Strength Indication (SSI). Third, the rate controller, which is at the sending side, needs in fact the SNR observed at the receiving side. The problem of communicating the SNR information back is addressed in the emerging 802.11h standard [12], but the standard has not been finalized yet. In any case, this will be always an open issue for the products that do not support this standard. Also, being aimed at 5GHz, it is not possible for 802.11h to be a successor for the standards like 802.11b and 802.11g that operate at 2.4GHz. This would require defining a new supplement. Another drawback of 802.11h is that the SNR-related information (referred as Link Margin there) is transmitted back by the help of an additional management frame. The latter will increase the MAC overhead, and can cause the SNR information to be delayed due to contention for the medium, and thus not delivered on time.

Most work on using SNR information for automatic rate control is based on simulation and does not consider the practical difficulties of obtaining good SNR estimates. It concentrates on the way in which the noisy and drifting SNR (problem 1) can be used to determine the correct rate setting [1, 13]. Holland et al. [6] do address the issue of how to communicate back SNR values (problem 3), but their rate selection algorithm still relies on a straight SNR threshold technique. Another approach is discussed in [3], where the assumption is made that the channel is symmetric, meaning that the SNR observed at either station is very similar for any given point in time. This assumption allows Pavon et al. to use the SNR of the last ACK frame as an indicator of the SNR at the other side, and to use it for selecting the rate of the next data frame to be sent. Pavon et al. avoid the issue of estimating the true SNR out of an SSI reading (problem 2) by continuously adapting a table that maps SSI into throughput for the four 802.11b data rates. The adaptation is necessary to accommodate for varying noise levels. However, their control algorithm does not utilize the full potential of SNR information, since the adaptation only takes place on retransmissions; hence, adapting upwards is not supported well. Also, they provide simulation results only; therefore it is not clear

how their algorithm will behave in practice. Finally, they do not discuss the dynamics of their approach, that is, how fast the rate setting reaches the optimal value for certain conditions, after these conditions were established.

### 3.3 Hybrid automatic rate control

Both the statistics-based and the SNR-based approaches have their advantages and disadvantages. The statistics-based approach gives robust performance and inherently maximizes the throughput in the long term. However, the main drawback is its slow response to changing link conditions, which can be a source of problems for real-time applications. The SNR-based rate control can respond very fast, but due to the uncertain and fluctuating relation between SNR information and BER of the link, it lacks stability and reliability. Therefore, a logical step forward is to combine the two approaches in a hybrid algorithm that will provide both robustness and fast response.

This paper describes such an algorithm and its implementation.

## 4 Practical implementation of hybrid CSI rate control

In this section we describe an advanced rate control algorithm that combines the advantages of statistics-based and SNR-based methods. The goal is to support streaming applications by limiting the packet delay and jitter as much as possible, even at the expense of throughput when necessary. This requires a hybrid approach, since under stable link conditions we want to provide a robust, high-throughput connection (statistics-based approach), but under volatile conditions induced by, for example, user movement, we want to avoid retransmissions by rapidly switching to a lower rate (SNR-based approach). Another important goal is to design a working system, so we must address the practical problems associated with SNR-based methods as discussed in the previous section. Results of our experimental prototype will be presented in Section 5.

The core of our hybrid algorithm is a traditional statistics-based controller. More specifically, it is a

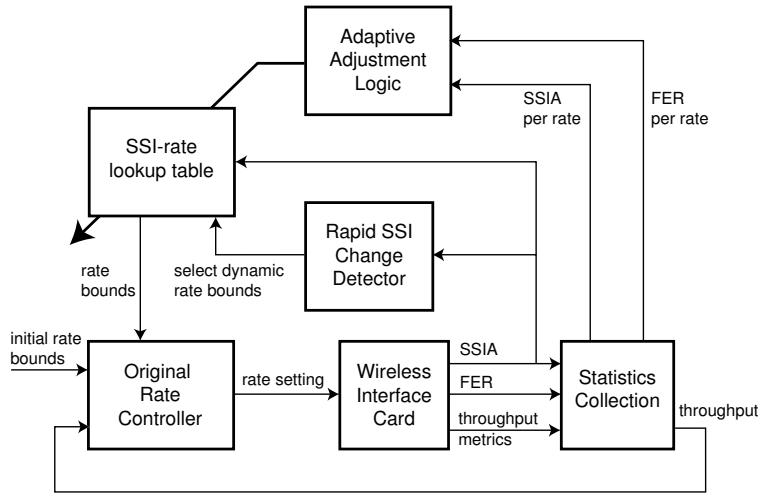


Figure 2: Structure of the hybrid rate controller.

throughput-based rate controller, which probes adjacent rates to determine if a rate switch is necessary (see Section 3.1.1). Figure 2 shows the complete structure of our hybrid rate controller; the throughput-based controller is part of the feedback loop depicted at the bottom of Figure 2, including the Wireless Interface Card and Statistics Collection component.

The decisions of the core controller can be overridden by a second feedback loop. This loop bounds the acceptable range of the Signal Strength Indication of the Acknowledged frames (SSIA) values for each rate, based on the specific knee in the Throughput-vs-SNR curve (cf. Figure 1). These SSIA bounds are implemented as a lookup table (SSI-rate lookup table component in Figure 2), which is indexed by the intended rate, given by the core controller. Such a table is presented in Table 1.

Rate (Mbit/s)		6	9	12	18	24	36	48	54
Low threshold (dB)	stable	7	9	11	13	15	18	22	25
	volatile	12	14	16	18	20	23	27	30
High threshold (dB)		17	19	21	23	25	28	32	35

Table 1: Sample SSI-rate lookup table

For example, if the core controller decides on a rate setting of 36 Mbit/s, and at the same time the last measured SSIA is 12 dB, then the rate that will be actually used is 12 Mbit/s. The bounds on the maximum rate are tightened when channel conditions change rapidly. This is done by addressing the threshold values marked as “volatile” in the SSI-rate lookup table, instead of the “stable” values that are used in slow-changing channel conditions. The rationale of this scheme is that it is better to play safe in volatile conditions, and switch to a lower rate to minimize packet jitter by reducing the probability of a frame loss. Rapid changes in channel conditions are detected by sensing the changes of consecutive SSIA readings; the component is called Rapid SSI Change Detector (RSCD) and is shown in the middle of Figure 2. To account for the drift of the SSIA readings, we employ an adaptive adjustment logic that updates the values in the lookup table according to the recent history of channel conditions. This approach is similar to that described by Pavon et al. in [3], and may be regarded as an auto-calibration loop.

The operation of the hybrid control algorithm will be detailed further in two parts. First, we will describe the extended rate controller, which comprises all components in Figure 2 except the adaptive adjustment logic. Second, we will discuss the auto-calibration loop, that is, the details of the adaptive adjustment logic.

The SNR-based extension of the rate controller is centered around the SSI-rate lookup table. This table contains three SSI thresholds per rate: two for stable conditions (high and low thresholds), and an additional low threshold for volatile link conditions. These thresholds describe for each rate setting the SSIA values that are allowed for its use. The low threshold used in stable conditions is referred to as the *absolute* low SSIA value that provides acceptable performance (defined as FER < 10%) under ideal circumstances. The low threshold used under volatile conditions is referred to as the *dynamic* low SSIA value. The RSCD decides which of the two low thresholds (absolute or dynamic) will be used for bounding the rate selection for the next frame. The detector observes the differences between three consecutive SSIA readings, comparing them only if they are taken within a certain, limited time period. If both differences have the same sign, and their sum exceeds a certain threshold, then the dynamic low threshold is used, since conditions are changing fast and consistently. Otherwise, the absolute low threshold for stable conditions is used. The RSCD has also a hold property, meaning that it will keep the indication for rapid change active for certain small amount of

time, even if this change is no longer detected. This feature improves the stability of the controller and is based on the assumption that link quality changes are caused by physical processes, which require at least some minimal time to take place.

The selection of the rate `RS_curr` of the next frame to be transmitted proceeds as follows:

```
once_per_decision_window() {
    RS_opt = find_opt_rate_by_stats();
    may_upscale = true;
}

for_each_packet() {
    RS_curr = probe_or_not(RS_opt);
    calculate_bounds();
    if (RS_curr > RUpBound)
        RS_curr = RUpBound;
    else if (RS_curr < RLoBound &&
            may_upscale) {
        RS_curr = RLoBound;
        try_upscale = true;
    }
    xmit_success = xmit_packet(RS_curr);
    if (xmit_success)
        update_SSIA_stats();
    if (try_upscale) {
        if (xmit_success)
            RS_opt = RS_curr;
        else
            may_upscale = false;
        try_upscale = false;
    }
}
```

```

    }
}

```

Here, `RS_opt` is the optimal rate as calculated by the core statistics-based algorithm at the end of each decision window. The upper and the lower rate bounds (`RUpBound` and `RLoBound`) are calculated from the SSIA thresholds in the lookup table and the SSIA from the previous packet (`prev_SSIA`), by the `calculate_bounds` procedure listed below. If the transmission is successful (indicated by the flag `xmit_success`), we also update the SSIA statistics by the `update_SSIA_stats` procedure. Note that when the selected rate (`RS_curr`) falls below the lower bound (`RLoBound`), we attempt to upscale by transmitting a frame at the `RLoBound` rate. If the transmission succeeds, we continue to use that rate. If it fails, we block additional attempts for the remainder of the decision window. This behavior is achieved by the help of the flags `may_upscale` and `try_upscale`.

```

calculate_bounds() {
    for (i=max_rate_index, i > 0, i--) {
        if (prev_SSIA >=
            SSIA_tbl[i].lo_thld[is_dyn])
            break;
    }
    RUpBound = RateTable[i];
    for (i=0, i < max_rate_index, i++) {
        if (prev_SSIA <= SSIA_tbl[i].hi_thld)
            break;
    }
    RLoBound = RateTable[i];
}

```

The low SSIA thresholds determine the upper rate bound for the current SSIA value. Likewise, the high SSIA thresholds determine the lower rate bound. When there are rapid changes in channel conditions, the SSIA

dynamics detector raises the `is_dyn` flag above, causing the algorithm to use the dynamic SSIA thresholds for calculating a stronger `RUpBound`.

Before we explore the work of the adjustment logic, lets first discuss how we can detect the situations when the SSIA thresholds need to be adjusted. Normally, a SSIA threshold should be positioned near to the right of the knee of the FER-SSI curve for the corresponding rate setting. Lets suppose that the threshold for rate  $k$  is too much to the left. Then there will be packets, attempted at  $k$ , and with SSIA greater then the SSIA threshold, which will experience many retransmissions before they are successfully received at the other side. This can be used as an indication that the SSIA threshold for rate  $k$  should be increased. On the other hand, if the SSIA threshold for rate  $k$  is too much to the right, then almost all packets, allowed for transmission at rate  $k$ , will be sent successfully at the first transmission attempt. So, this would be our indication that the SSIA threshold for rate  $k$  should be decreased.

For each rate, two counters are kept, which are updated on per-packet basis. The first counter, `lo_SSIA`, keeps the number of the packets, whose transmission resulted in indication that the threshold for this rate should be increased. Similarly, the second counter, `hi_SSIA`, keeps the number of the packets, whose transmission resulted in indication that the threshold for this rate should be decreased.

At the end of each decision window, the update of the thresholds for all rates that were used in this decision window, is performed as follows:

```
Adjust_SSIA_thresholds() {
    if (#packets < MIN_PKT_NUM)
        return;
    for (i=0, i < max_rate_index, i++) {
        if (SSIA_tbl[i].lo_SSIA / #packets > a)
            SSIA_tbl[i].lo_thld += c;
        else if (SSIA_tbl[i].hi_SSIA / #packets > b)
            SSIA_tbl[i].lo_thld -= d;
        SSIA_tbl[i].hi_thld = SSIA_tbl[i].lo_thld + e;
    }
}
```

```

        SSIA_tbl[i].lo_thld_dyn = SSIA_tbl[i].lo_thld + f;
    }
}

```

Here,  $a$ ,  $b$ ,  $c$ , and  $d$  are coefficients that determine the accommodation properties of the adaptation algorithm. Coefficients  $a$  and  $b$  are thresholds, used together with `lo_SSIA` and `hi_SSIA` as an integrating detection mechanism to determine the need and the direction of SSIA threshold adjustment. Thus,  $a$  and  $b$  determine the filtering properties of this detection mechanism. Coefficients  $c$  and  $d$  determine the magnitude of a single adjustment to the SSIA thresholds per decision window, in the event we detect that it is necessary to make an adjustment during this window. These coefficients can be compared to the D (differential) component in a PID controller. In the current implementation,  $e$  and  $f$  are constants, but in the future, they will represent functions. The  $e$  function accounts for the magnitude of the drift in the FER-SSI relations. The  $f$  function determines how pessimistic the algorithm will be in the case of rapidly changing channel conditions, thus influencing the responsiveness of the rate controller.

Tuning the coefficients is done by experimenting with different sets of values, and selecting the ones that produce the best performance results, yet yield a stable operation under all operating conditions that we have tested. Initial analysis shows that the algorithm is most sensitive to the choice of  $a$ ,  $b$  thresholds, but fortunately determining their values is subject to the least variance in our experiments. Currently, the values used are  $a = 0.1$ ,  $b = 0.8$ ,  $c = 1$ ,  $d = 1$ ,  $e = 10$  and  $f = 5$ . Part of our future work is to perform a thorough sensitivity analysis, and to establish a firm, model-based procedure to adjust the coefficients, depending on the particular case the hybrid algorithm is used for.

After updating the thresholds for the rates used during the decision window, the thresholds for the unused rates may need to be updated as well to preserve the monotonic value growth in the lookup table. An example of a specific update situation is shown in Figure 3, where the thresholds for rate indexes 1 and 4 have already been updated according to the procedure outlined above. Next, the thresholds for the other rate indexes are checked. Rate index 2 is found to violate the monotonicity requirement, and its value is increased to that of the preceding rate index 1. No further adjustments are needed. To handle potential

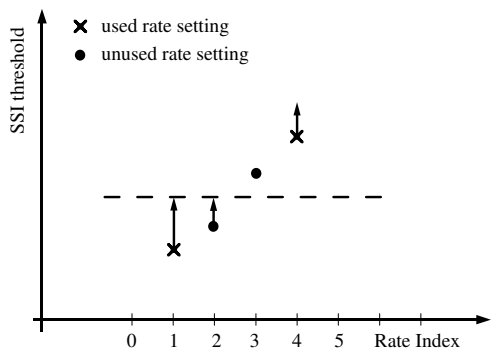


Figure 3: Resolving specific cases for SSIA thresholds update.

conflicts between threshold updates, the adjustments are performed from the lowest rate to the highest rate.

## 5 Experimental evaluation

We have implemented our hybrid algorithm on real hardware and carried out a number of experiments executed in real-time. In this section, the traffic we use is synthetic, and the main goal is to explore different parts of the algorithm, and to give assessment for their contribution to the performance. A real scenario evaluation will be made in the next section.

### 5.1 Experimental setup

Our experimental setup consists of two laptops, both equipped with a Proxim Skyline 802.11a wireless network interface card supporting eight different data rates, ranging from 6 to 54 MB/s. These cards are based on the Atheros 802.11a AR5000 chipset. The advantage of using this chipset is that it only implements the lowest MAC functionality (e.g., basic medium access, retransmissions control, etc.) and leaves the remainder to the device driver. This provides us with the flexibility to modify the original MAC protocol, and even to replace the automatic rate controller with our own hybrid approach.

The laptops are running Linux, and we developed the AR5000 device driver ourselves since only a

Windows version was supported by Atheros. The device driver includes our hybrid automatic rate controller discussed in the previous section. For experimentation purposes the driver can be instructed to disable the SNR-based components so that we could measure the performance of the original throughput-based controller developed by Atheros. We modified the popular Netperf 2.2 tool to set the low-level MAC parameters exported by our device driver: maximum number of retransmissions, rate setting, Tx power, etc. We also used Netperf to generate data traffic and to collect statistics about the raw performance of the wireless connection between the two laptops in a standard office environment (with concrete walls).

## 5.2 SSIA reasoning

To obtain basic insight into the performance of the wireless network cards, we conducted a first experiment in which we placed one laptop at different positions in the corridor. At each position we exercised all eight 802.11a rate settings logging the throughput reported by Netperf and the signal strength (of the acknowledgements) as reported by the network interface. Figure 4 shows the relationship between signal strength, rate setting (modulation scheme), and throughput for our experimental setup. In addition to the raw measurements, Figure 4 includes a smoothed<sup>1</sup> curve for each rate setting to better visualize the cliff behavior. The infeasibility of directly implementing a SNR-based rate control algorithm can be seen for example by looking at the results for 24 Mbit/s. Note that the data points represent averaged values over time. But still, some of the points (like those at a SSIA of 23 with throughput below 2Mbits/s) suggest that the cliff for that rate should be at about a SSIA of 23, while others on the left determine the switch over point to be around 15. Thus, straightforward SNR-based control will lead to unstable, and sometimes erratic behavior.

The SSI readings that we obtained from the chipset ranged in practice from 0 (complete loss of connection) to about 70 (cards next to each other). Unfortunately, the documentation about the AR5000 chipset does not specify the unit of these measurements. Therefore we performed a series of open-space experiments in which

---

<sup>1</sup>To smooth the data, we first excluded values that violate the monotonic increase of the curves. Then, we interpolated the data to obtain regularly spaced values. Finally, we ran the data through a low-pass FIR filter.

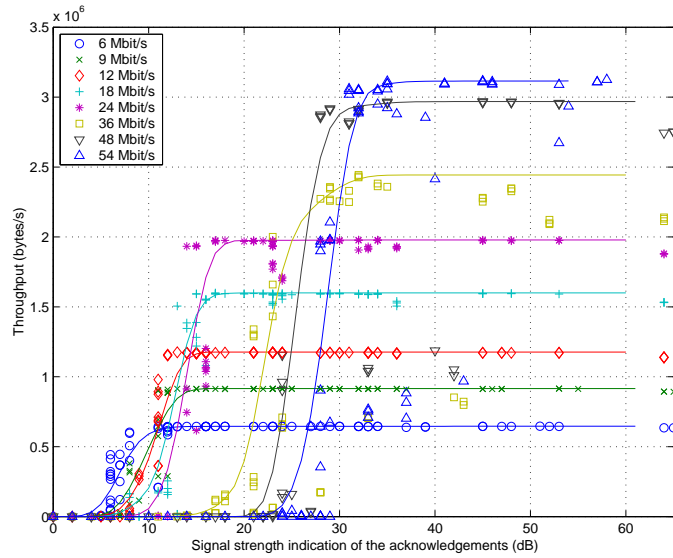


Figure 4: Throughput vs. SSI of acknowledgments (SSIA).

we varied the distance, which revealed that the obtained SSI values are in dB over some reference point. This reference point is most likely linked to the noise floor of the receiver, thus we end up getting ‘SNR-related’ information. Our experiments show that the noise floor of the receiver is recalibrated periodically, which contributes to the drifting in time of the throughput-SSI curves. It is important to note here that, since we use a self-adapting algorithm, the reference point of the SSI readings does not influence the operation of the algorithm. This is provided, of course, that the reference point does not change too fast (within a second), which is the case in our experimental setup.

Figure 4 clearly shows that in reality, the obtained SSI feedback is quite noisy, especially around the edge of each cliff. In addition, the noise level for more complex modulation schemes (i.e., higher data rates) is quite high, even on the flat part of the curve. For example, at an SSI of about 42, the throughput of the three top rates suddenly plummets to a rate of just 1 Mbit/s. This is most probably due to complete Rayleigh fading of some of the OFDM carriers used in the 802.11a radio at that particular position in the corridor.

A second observation that can be made from the curves in Figure 4 is that, in our experimental setup, it does not make sense to use the 9 Mbit/s rate setting at all, since this setting does not outperform any other rate setting for any SSIA value. Although we did not investigate if this holds for any possible scenario, all our experiments, and also simulations by [9] show that the 9 Mbit/s rate setting is generally of no use.

Finally we like to point out the throughput reduction caused by MAC and PHY layers overhead. This overhead is due to extra information added to the data payload at MAC and PHY layers (in the form of headers, preambles, etc.), and the fact that the information is transmitted at a fixed (low) rate. This overhead is part of the 802.11 standard [10, 11], and can cause significant deviation from the PHY raw data throughput [7], especially at high rate settings. At the 6 Mbit/s setting, we logged a Netperf throughput of 4.9 Mbit/s, hence, the MAC/PHY overhead is limited to 18%. At the 54 Mbit/s setting, the Netperf throughput is limited to just 23.8 Mbit/s, hence, the overhead has grown to 56%. Note that the actual throughput, and consequently the MAC/PHY overhead, depends on the frame size used. In our case, the packets are 1024 bytes long, and we did not use packet fragmentation, so the frame size is equal to the packet size.

We conducted a second experiment to verify the assumption that the communication channel is symmetric, so that we can use the signal strength of the acknowledgements (SSIA) observed at the sender instead of relaying the true signal strength (RxSSI) observed at the receiver. The experiment involved a person walking around in a random fashion with one of the laptops (the other laptop remained stationary), and recording both the SSIA at the transmitting side and the SSI at the receiving side for each packet. Figure 5 shows how the signal strength observed at the receiver changed during the 140 sec experiment (packets are streamed at a rate of 100 per second). Note that during stable conditions (standing still), the noise in the SSI is limited to  $\pm 2$  dB; while moving, however, the fluctuations are much larger (up to  $\pm 17$  dB).

Figure 6 provides a histogram of the differences between the signal strength observed at the sender (SSIA) and receiver (RxSSI). Even though the SSIA and RxSSI readings are not captured from the channel at the same time (the ACK follows the DATA packet), it is quite clear that the SSIA and RxSSI values are strongly correlated. For 74% of the packets, the difference falls within the stationary  $\pm 2$  dB noise range, and even

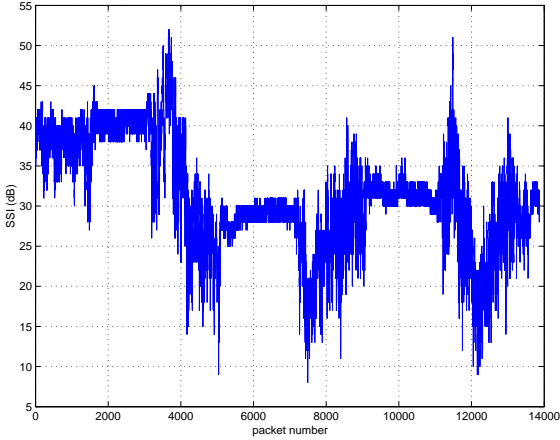


Figure 5: SSI at the receiving side (RxSSI), random walk.

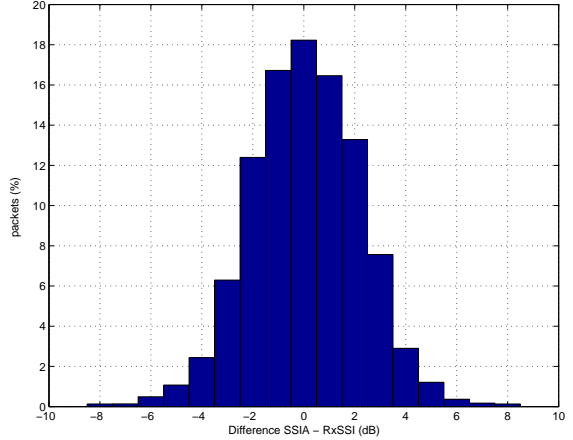


Figure 6: Histogram of the difference SSIA - RxSSI.

the largest differences are in line with the noise during movement. Therefore, from a control perspective, we can safely use SSIA instead of RxSSI, thus avoiding the problem of relaying the SSI at the receiver back to the rate controller operating at the sender.

### 5.3 Step response functions

In the third experiment we compared the original throughput-based rate controller supplied by the manufacturer of the chipset (Atheros) with our hybrid controller. To allow for a head-to-head comparison under the same conditions, we decided not to use a real-life scenario with a walking person, since repeating the experiment (i.e., regenerating the exact same channel conditions) would be close to impossible. Instead, we measure the Step Response Function (SRF) in an artificial setting to determine the responsiveness, robustness, etc. of the rate control algorithms.

In general the SRF is used to study how some control system reacts when a certain input parameter (in our case, the SSI) changes instantly, that is, in one step from one value to another. In our SRF experiment

we placed one of the laptops in a microwave oven, started streaming data to the other laptop, and then closed the door rapidly, causing the SSI to swiftly drop from about 35 (good) to about 10 (bad), because the shielding of the microwave oven blocks almost all radio waves in the 5 GHz band. After about 3 seconds the door was opened again, generating a second step in the channel conditions (from bad to good). Since the microwave door was opened and closed manually, the exact channel conditions (i.e, SSI readings) show small variations between experiments (cf. figures 7 and 8). Nevertheless, we can reproduce the SSI steps with enough accuracy to show the (large) differences between the throughput-based and hybrid rate control algorithms. To stress these differences we impose a traffic load that corresponds to a demanding streaming-video application; Netperf was instructed to send 1024 bytes long packets at a rate of 100 packets/s, resulting in an effective data rate of 800 Kbit/s.

Figures 7, 9, and 11 show the SSIA readings (input), generated rate settings (output), and induced packet latencies (performance), respectively for the original throughput-based rate control algorithm. Figures 8, 10, and 12 show the same for our hybrid algorithm. According to expectations the throughput-based controller cannot accommodate fast enough to the step down in channel conditions (around packet number 1000) because of the 1 second decision window. When it does respond (around packet number 1080), the controller decides to step down one rate setting (from 54 to 48 Mbit/s), which is too little too late. All packets transmitted during the closing of the microwave get lost; the latencies for these packets are labelled  $\text{inf}(\text{inity})$  in Figure 11. As a consequence, no acknowledgements are received, causing a gap in the SSIA readings shown in Figure 7 and, hence, a lack of throughput statistics leading the controller to maintain the current (way too high) setting. This situation continues until the channel conditions improve after opening the microwave door, and the controller can finally decide to step down further (from 48 to 36 Mbit/s) around packet number 1380. Our hybrid controller, in contrast, responds almost immediately to the drop in channel conditions and overrules the decisions of its internal throughput-based controller based on the SSIA thresholds. Consequently, almost all packets are transmitted at the appropriate rate and service continues smoothly. When the conditions improve (around packet number 1290) the hybrid controller is also quick to respond and switches to higher rate settings.

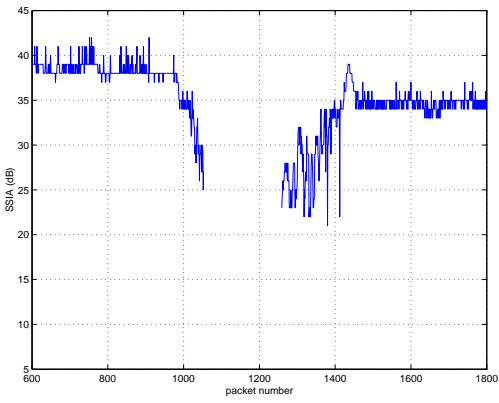


Figure 7: SSIA for the statistics-based algorithm.

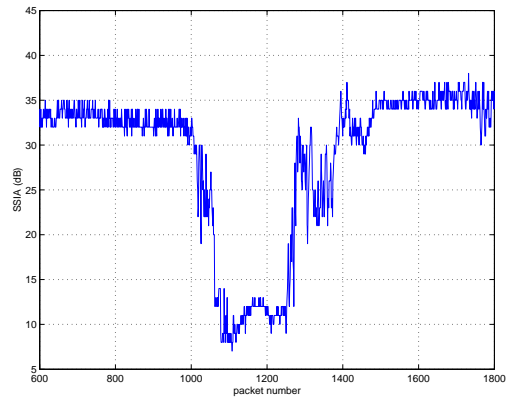


Figure 8: SSIA for the hybrid algorithm.

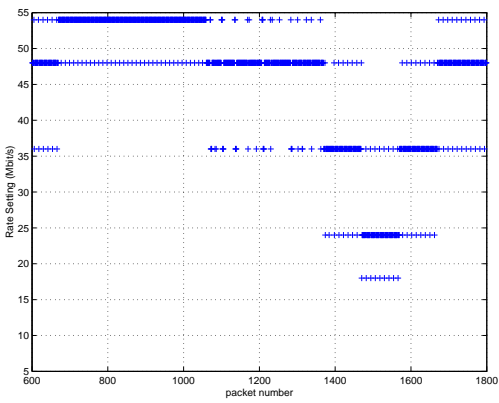


Figure 9: Step response function - statistics-based algorithm.

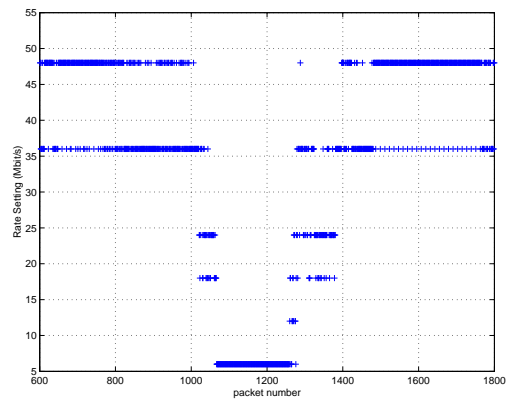


Figure 10: Step response function - hybrid algorithm.

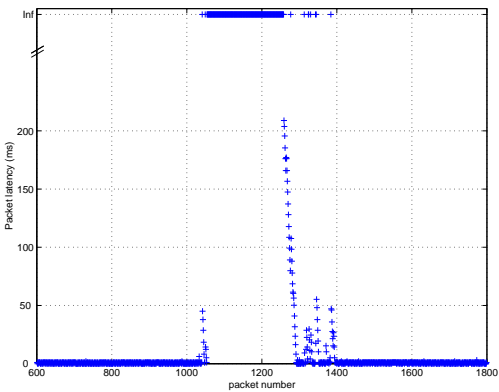


Figure 11: Packet latency - statistics-based algorithm.

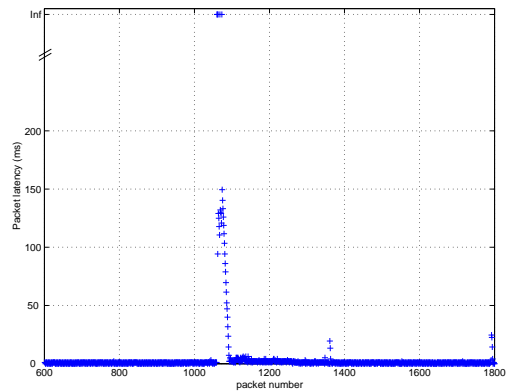


Figure 12: Packet latency - hybrid algorithm.

The step-response function in Figure 9 shows the workings of the throughput-based controller in quite some detail. First of all, we can clearly see the 1 second decision window in the rate selections, for example, packets numbered 1380 to 1680 are grouped into three windows with rates 36, 24, and 36 Mbit/s respectively. Also, the probing at adjacent levels (1 in 10 packets) is visible as the light bars up and/or below the dark bar of the selected rate. Figure 10 shows that the hybrid algorithm does probe at adjacent levels too, but that quite often the SSIA-based thresholds overrule an up-probe. As an example, consider the time frame from packet number 1500 to 1750. In this period, the majority of the packets is sent at the 48 Mbit/s rate, some are sent at the adjacent lower rate (36 Mbit/s), but none at the higher rate (54 Mbit/s). This behavior follows from the data in Figure 4, which shows that around 35 dB the 54 Mbit/s throughput curve is still rising, signalling unstable conditions.

The packet latency, as reported in figures 11 and 12, is measured as the time between the moment the device driver inserts a packet in the send queue, and the moment that the chipset starts the actual, physical transmission. The latter time stamp is returned (by means of an interrupt) to the device driver when the acknowledgement packet is received. The chipset automatically takes care of retransmissions. If the maximum number of retries (10 in our case) does not result in a successful transmission, the packet is dropped, and an interrupt is generated to inform the driver. The device driver then marks the latency for the dropped packet as infinite. Under normal conditions the send queue is (almost) empty, but when retransmissions are issued, the number of packets in the queue grows. When the control algorithm then switches to the “right” rate, the packets in the queue are transmitted back to back until the queue becomes empty again. This effect is clearly visible in the latency graphs: after a sequence of dropped packets, latency jumps and then rapidly declines to (near) zero. Streaming applications are quite sensitive to latency and usually put a strict requirement on the maximum latency for any packet.

The bottom line is that, from the point of view of a streaming application, the difference between the two automatic rate controllers is huge. With the original throughput-based controller, the stream is halted for about 2 seconds (205 packets are lost), causing a complete outage in service to the user. With the hybrid controller, the user may experience some service degradation depending on the application’s ability to cover

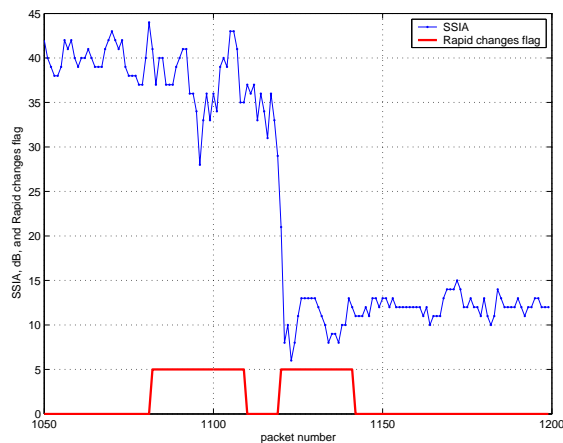
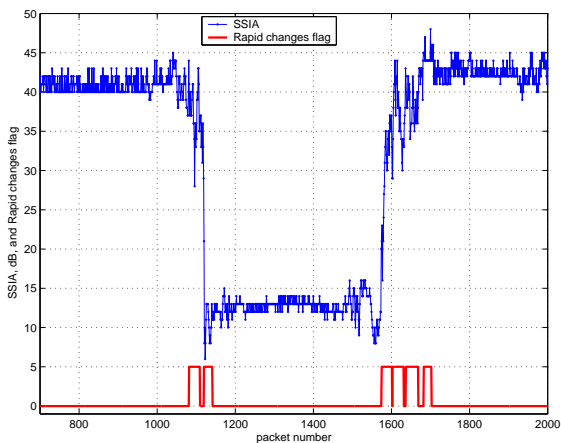


Figure 13: Dynamic detection algorithm performance. Figure 14: Dynamic detection algorithm performance - zoomed.

up the few (5) dropped packets. In the case of a streaming video application, the difference is between losing connection and observing some artifacts in a few frames; a significant difference indeed.

## 5.4 Rapid SSI Change Detector (RSCD)

We expect that delays and packet losses will be quite critical for multimedia applications (leading to artifacts in a video and dropouts/hickups in video/audio, see Section 6), therefore it is very important that we avoid delays and losses as much as possible. As it was already discussed, RSCD detects the moments when the link conditions change drastically. Then it instructs the rate controller to become more conservative, effectively providing the necessary safety margin in time, in case the algorithm has to react to rapidly declining link conditions. Here, we evaluate the performance gain that this detection circuit adds, by comparing the results of the work of the rate controller with, and without, RSCD.

Figures 13 and 14 show the moments that RSCD determines as “turbulent”, i.e. those in which there is a rapid change in the link conditions.

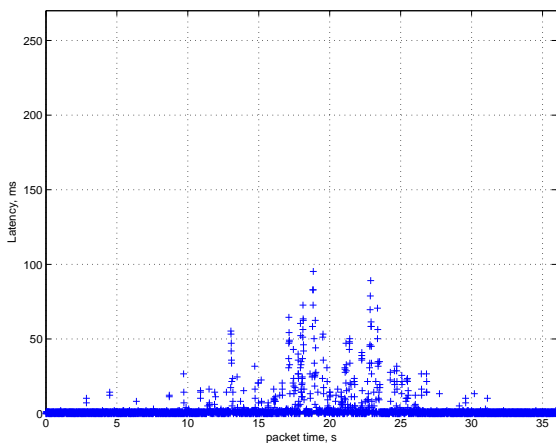


Figure 15: Packet latencies with dynamic detection algorithm.

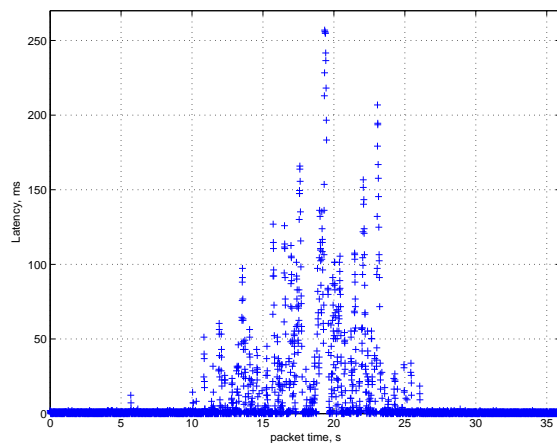


Figure 16: Packet latencies without dynamic detection algorithm.

By comparing different situations, we can tune the parameters of RSCD so that it is sensitive enough to identify the really turbulent periods, and in the same time, not to produce (too many) false alarms.

The results for the comparison of the performance of the rate control systems with, and without RSCD, is shown in figures 15 and 16.

The benefit of using RSCD is clearly seen, as the peak latency for the system with RSCD barely reaches 100 ms, while the peak latency for the system without RSCD exceeds 250 ms. Also, with RSCD, the lost packets are 44, opposed to 126 lost packets in the case when RSCD is not used.

## 5.5 SSI thresholds adaptation circuit (STAC)

We have already discussed that STAC is responsible for adjusting the thresholds in the SSI lookup table, following the shifts of the FER-SSI curves. A good way to evaluate the operation of STAC is to set the initial SSI thresholds to some “invalid” values and observe how STAC modifies them. The results are shown in figures 17 and 18. The first one shows the adaptation process when the thresholds are set to extremely low

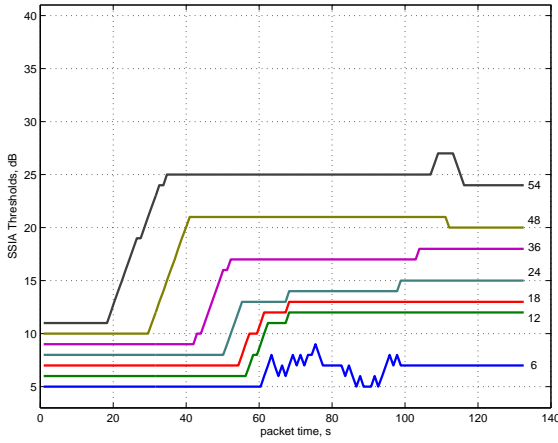


Figure 17: Adaptation of SSIA thresholds with low initial values.

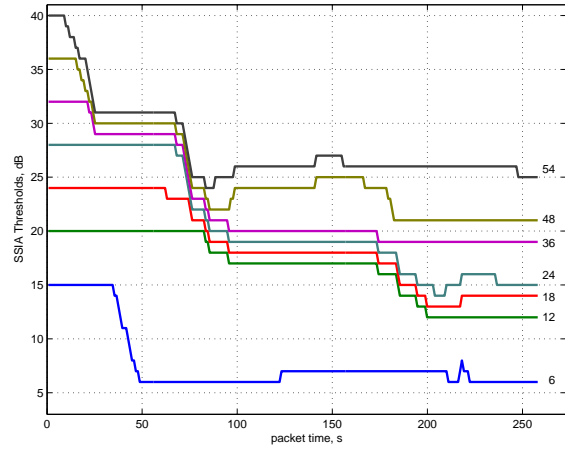


Figure 18: Adaptation of SSIA thresholds with high initial values.

values, the second, respectively, shows the accommodation when the thresholds were initially high. As it can be seen, in both cases the thresholds reach values that are close to the average “standard” values, which for the rates from 6 to 54 Mbits/s are 7, 12, 14, 16, 18, 22 and 25, respectively. The average “standard” values were determined by getting the FER-SSI curves under different environmental conditions, and calculating the mean for the SSI values, for which FER was equal to 50% (the middle of the knee of the curves). The curves in figure 17 do not reach exactly the same values as these in figure 18, because of the drift of the FER-SSI curves in time, and because of noise in the SSIA acquisition.

It can be noticed, that in both cases the full accommodation takes a significant amount of time. However, on the one hand these are extreme cases, and in real situations the initial values would be much closer to the optimal ones. On the other hand, the fluctuation speed of the optimal values is rather low, so STAC will have in practice enough time to accommodate.

The factors that influence the speed of adjustment of the thresholds include the change of the link conditions, the packet rate and burst degree, and many others. Therefore, the discussion about these factors

goes out of the scope of this paper, and will be presented elsewhere.

## 6 Real video-streaming scenario evaluation

In this final experiment we will try to validate the previous experiments that use synthetic traffic, with a realistic video streaming application in an office environment with one of the two laptops moving around.

Because of the nature of video codecs like MPEG1/2 and H.263, lost packets can have disastrous effects on the image quality. To decode a certain (video) frame, the decoder depends on the previous frame being decoded correctly. When lost, a single packet which contains information of part of a video frame, will generate effects on the next frames as well. For any streaming transmission, for example video-conferencing or just television, all packets should arrive in time, otherwise they are basically useless. We assumed a latency constraint of 500 (ms) between transmission and reception. This means we should prevent to lose packets, since there is no time to use end-to-end (TCP-level) retransmissions. The link or MAC-level retransmissions, on the other hand, are an effective way to deal with packet losses.

In this experiment we will compare the decoded video quality for different algorithms. The setup of our experiment is as follows: One of the laptops is on a fixed position on a desk. The other laptop is carried out the room and then up and down through the hallway a few times, and then returns to the initial position next to the other laptop. This experiment takes two minutes. By walking up and down the hallway we ensure that the conditions are continuously improving or deteriorating, so we can inspect the behavior of the rate control algorithm at different circumstances. During the whole experiment the first laptop was streaming video to the second laptop which decoded and recorded the received video. Since we are evaluating a real-time streaming and displaying application, we discarded all packets that arrived when in fact the next video frame should already be decoded.

Each experiment is run three times to be able to correct for slightly varying conditions during the experiments. The transmission consisted of a H.263 encoded stream of the “carphone” sequence at QCIF format. It is a representative video often used for comparing video coders. Since we are just evaluating the

effects of the rate-control algorithm on the number of packet losses, the produced video rate is lower than the lowest rate setting (6 Mb/s).

Afterwards, we have investigated the quality of each received stream. A classical and easy measure for image quality is the peak-signal-to-noise-ratio (PSNR) measure. The downside of this measure is that it not necessarily corresponds to the human perception and that it is not well-suited for moving pictures where sometimes frames are missing. To overcome this shortcoming we employed two measures: 1) average PSNR: The average of the PSNR's of all frames. For each missing video frame, the last correctly received frame is taken in place<sup>2</sup>. 2) human perceptual quality. To obtain a perceptual measurement we have shown each received video to fourteen people who had to give a mark between 0 and 5 (0=bad, 5=good), of course without telling to which case this recording belonged. Both measures together should give a good indication of the effects of losing packets on the quality of the video.

We have compared the statistics based algorithm and our hybrid algorithm with a perfect transmission and with a reference algorithm. Table 2 shows for the following cases, the packet loss ratio the PSNR quality and the perceptual quality measure on a scale of 0 – 5.

1. Perfect transmission: In this fictitious algorithm, we simulated that there were no lost packets at all, resulting in the highest quality possible in this setup.
2. Reference algorithm: the rate setting is fixed at the lowest value, therefore with the lowest packet loss probability. This represents the best any rate control algorithm can do under these circumstances.
3. Paranoid Hybrid algorithm: uses the hybrid algorithm, but it switches to lower rates at a slight indication of deteriorating channel conditions, hereby trying to prevent losing packets. This is achieved by shifting upwards the SSIA thresholds in the SSI-rate lookup table. By limiting the number of attempts per decision window, the algorithm is made more conservative to go to higher rates, when the link conditions are good.

---

<sup>2</sup>This also happens visually with a real-time decoder. We did not employ any advanced error concealment techniques at the video decoder, because we want to have a clean comparison between the rate control algorithms

4. Hybrid algorithm: our hybrid rate controller as evaluated in the previous sections.

5. Statistics based algorithm.

Algorithm	packet loss	perceptual quality	average PSNR (dB)
1. Perfect	0.00%	5.0	37.34
2. Reference	0.08%	4.2	36.96
3. Hybrid – paranoid	0.15%	3.0	36.59
4. Hybrid	0.97%	1.3	34.73
5. Statistics-based	7.01%	0.4	29.33

Table 2: Packet loss and PSNR measurements and perceptual quality rating for five different algorithms

The results in Table 2 show that the statistics based algorithm generated seven times as many packet losses than our hybrid algorithm. The video quality for the hybrid algorithm is therefore 5.4 dB higher than for the statistics based algorithm. The perceptual rating confirms this difference. As can be seen from the perceptual measurements, even low packet-loss ratios already give significant lower qualities. This is due to the propagation of errors in consecutive frames. Therefore we introduced a more paranoid version of the algorithm. This version goes to greater extent to prevent losing packets by switching to lower rates at an earlier stage and being hesitant to increase the rate when the conditions improve. The result was that fewer packets were dropped, and the achieved quality was closer to the reference algorithm. On the other hand we should remark that on average, this paranoid algorithm will as a result also obtain lower average rates than the normal hybrid algorithm, under the same circumstances. This is nonetheless not visible in our experiments.

## 7 Conclusions and future work

In this paper we addressed the problem of supporting real-time streaming applications over a wireless link. The strict latency requirements of streaming applications and the fluctuating link conditions require careful handling in all layers of the protocol stack. At the lowest level, the IEEE 802.11 standard includes several tuning parameters, of which the transmission rate (selected per packet) is the most important one. Standard device drivers for today's wireless 802.11 products are based on gathering statistics and, consequently, adapt rather slowly to changes in conditions. Using signal strength (SNR) information is appealing because it provides instant information about the channel conditions. However, since SNR readings are quite noisy, have a wide variance, and are unstable, SNR cannot be used in practice to directly select the rate.

We have designed and implemented a hybrid rate controller that uses the available SNR information as a safeguard. This hybrid controller is built around a traditional statistics-based rate controller. The SNR information is used to limit the range of feasible settings from which the core controller can choose. This prevents the transmission of packets at a too high rate when channel conditions suddenly change for the worse. In a controlled experiment, we measured the step response function of both the statistics-based rate controller provided by the manufacturer (Atheros) of the 802.11 chipset that we employ, and that of our hybrid rate controller. The performance difference is large: whereas the statistics-based algorithm incurs a total dropout due to its slow response, the hybrid algorithm adjusts almost instantly and loses only a few packets. We conclude that SNR information can be used to the advantage despite the practical problems of the noise and drift usually associated with it.

Although the gain of our hybrid algorithm with respect to the statistics based algorithm, is large in terms of packet losses and for instance video quality, the hybrid algorithm can still be improved. By being more paranoid, more packet losses can be prevented. However, there is clearly a tradeoff between the number the packet loss paranoia and bandwidth. We plan to investigate this trade-off more thoroughly.

We also plan to investigate the benefits of including other parameters to be controlled (like packet fragmentation and Tx power). We will also study the possible use of inter-layer QoS negotiations, so that

the application will be able to influence the algorithm controlling the radio parameters. Finally, to test our approach in more realistic circumstances, we will investigate more-complex real scenarios with more than two nodes.

## References

- [1] K. Balachandran, S.R. Kadaba, and S. Nanda. Channel quality estimation and rate adaptation for cellular mobile radio. *IEEE Journal on Selected Areas in Communications*, 17(7):1244–1256, July 1999.
- [2] B.E. Braswell and J.C. McEachen. Modeling Data Rate Agility in the IEEE 802.11a WLAN Protocol. In *OPNETWORK 2001*, March 2001.
- [3] Javier del Prado Pavon and Sunghyun Choi. Link adaptation strategy for IEEE 802.11 WLAN via received signal strength measurement. In *IEEE International Conference on Communications, 2003 (ICC '03)*, volume 2, pages 1108–1113, Anchorage, Alaska, USA, May 2003.
- [4] I. Haratcherev, K. Langendoen, I. Lagendijk, and H. Sips. Application-Directed Automatic 802.11 Rate Control: Design Rationale and Preliminary Results. In *ASCI 2003*, pages 56–60, Heijen, The Netherlands, June 2003.
- [5] Martin Heusse, Franck Rousseau, Gilles Berger-Sabbatel, and Andrzej Duda. Performance anomaly of 802.11b. In *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*, pages 836–843, San Francisco, California, USA, April 2003.
- [6] Gavin Holland, Nitin H. Vaidya, and Paramvir Bahl. A rate-adaptive MAC protocol for multi-hop wireless networks. In *ACM MOBICOM'01*, Rome, Italy, July 2001.
- [7] Jangeun Jun, Pushkin Peddabachagari, and Mihail L Sichitiu. Theoretical Maximum Throughput of IEEE 802.11 and its Applications. In *Second IEEE International Symposium on Network Computing and Applications*, pages 249–256, Cambridge, Massachusetts, USA, April 2003.

- [8] A. Kamerman and L. Monteban. WaveLAN-II: A High-Performance Wireless LAN for the Unlicensed Band. *Bell Labs Technical Journal*, pages 118–133, Summer 1997.
- [9] Daji Qiao, Sunghyun Choi, and Kang G. Shin. Goodput analysis and link adaptation for IEEE 802.11a wireless LANs. *IEEE Transactions on Mobile Computing*, 1(4):278–292, 2002.
- [10] IEEE standard 802.11. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 1999.
- [11] IEEE standard 802.11a supplement. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. High-speed Physical Layer in the 5 GHz Band, 1999.
- [12] IEEE standard 802.11h supplement. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Spectrum and Transmit Power Management Extensions in the 5GHz band in Europe, 2003.
- [13] T. Ue, S. Sampei, N. Morinaga, and K. Hamaguchi. Symbol rate and modulation level-controlled adaptive modulation/TDMA/TDD system for high-bit-rate wireless data transmission. *IEEE Transactions on Vehicular Technology*, 47(4):1134–1147, November 1998.
- [14] A.J. van der Vegt. Auto Rate Fallback Algorithm for the IEEE 802.11a Standard. Technical report, Utrecht University, 2002.