ET4394 Rate Control Assignment

Pradhayini Ramamurthy (4180437) and Hans Okkerman (4290453)

***Abstract*— The 802.11ac standard introduced dynamic rate selection (MCS) on a channel at the MAC layer, allowing NICs to achieve real-time, optimal use of the channel. The standard, however, did not specify algorithms that dictate the manner in which the MCS rates ought to be chosen, leaving room for targeted and optimized algorithms. This document details the efforts of the WN_2 group towards the formulation of such an SNR-based algorithm for optimum throughput and minimum packet error rate.**

I. Introduction

The final assignment for the Wireless Networking course, EE4394, of the Delft University of Technology is to simulate and analyze a given dynamic rate control algorithm for an IEEE 802.11 data link using Matlab, and to propose an improvement over the given algorithm. The rate control algorithm under study operates on $VeryHighThroughtput(VHT)$ channels and changes the modulation and coding scheme (MCS) based on the received SNR of previous packets in an attempt to achieve the highest data throughput possible over a channel with a changing SNR.

The given example algorithm $TransmitRateControlExample$ works as follows: When a packet is to be transmitted, the algorithm first looks at the SNR of the previous packet. If this SNR is above a certain threshold value plus a given sensitivity constant, the coding rate is increased. If it is below another threshold value plus a sensitivity constant the coding rate in decreased. If both or neither of these conditions are met, the coding rate remains unchanged. The new packet is then transmitted using this selected coding rate.

In this paper another algorithm is introduced that is intended to improve the data throughput of a given channel when compared to the original. In addition, a simplified version of the $BanditLink$ algorithm [2] is implemented, and a performance comparison is done. The rest of this paper is structured as follows: First the used simulator will be explained. Next, the workings of the proposed algorithm will be covered. Following this, the $BanditLink$ algorithm and its implementation are discussed. After that, simulation results will be provided over several channel types. Finally the results will be compared to those of the original algorithm and conclusions will be drawn.

II. Channel parameters

The provided simulation takes several input parameters that define the channel characteristics and limitations. These parameters will consequently have a strong effect on how a rate control algorithm will perform. Of these, most are fixed or default parameters and only three are varied for the purpose of this report's simulation and performance estimation. The fixed parameter set includes the following:

1) $APEPLength$, which in turn gives the same $A-MPDU$ frame aggregation for all generated packets.
2) $NumTransmitAntennas$ and $NumReceiveAntennas$ configured to 1, thereby indicating a single user channel not utilizing any MIMO technology.
3) SNR parameters that produce a random, time-varying SNR for each generated packet.

The three variable parameters, which influence the algorithms' behavior and performance, will be covered next.

A. Bandwidth

The simulation uses Matlab's $wlanTGacChannel$ as its channel object. One of its inputs is the channel bandwidth. This bandwidth can be either $CBW20$, $CBW40$, $CBW80$ or $CBW160$, which correspond to bandwidths of 20MHz, 40MHz, 80MHz and 160MHz respectively. Certain combinations of bandwidth values and MCS modes result in errors in Matlab. For example, the combination of $CBW20$ with an MCS of 9 or higher is not supported. As such, for a bandwidth of 20MHz an additional check will be placed in the algorithms to limit the maximum MCS to 8 for those cases.

B. Delay model

Another input of Matlab's $wlanTGacChannel$ is the delay profile model. This model, which ranges from $Model-A$ through $Model-F$ controls the amount of reflections, delay spread and whether or not there exists a Line Of Sight (LOS) link based on the distance between the transmitter and receiver. Higher letters indicate more reflections and delays.

C. Distance

The distance input specifies the distance between the transmitter and receiver. This determines the path loss of the transmission, as well as whether or not the link between transmitter and receiver is line of sight depending on the chosen delay model, as mentioned above.

III. Proposed algorithm

Several different changes to improve the performance of the the original algorithm were explored. This section will briefly cover the workings of these new algorithms.

A. Moving average SNR

A first change to the algorithm was to add the history of the channel to the decision making process. Instead of only looking at the SNR of the previous packet, the algorithm now looks at the previous n packets, where n is an integer value greater than 1. Next, it finds the differences in SNR between sequential packets. These differences are then averaged to find a rough trend of how the SNR is changing. The coding scheme for the next packet is then changed accordingly based on whether the average of the previous SNRs plus the expected change in SNR is above or below a given threshold. By doing this the algorithm is expected to be slower to respond to quickly changing channel quality, but also less likely to overreact to short lived spikes in the SNR.

B. Weighted average SNR

The previously mentioned moving average algorithm could be extended by adding weights to the different places in history. By assigning larger weights to more recent changes in SNR and lower weights to changes that happened longer ago, the algorithm could be fine tuned for optimal performance. This way a balance could be found between being slow to react to changing situations and making wrong decisions based on short spikes in SNR.

IV. The $BanditLink$ algorithm

In order to fully understand the implications and applications of rate control algorithms, a brief literature survey on existing work was undertaken. Based on a broad classification of the most popular algorithms presented in [1] the sample algorithm $TransmitRateControlExample$ that serves as the starting point for our work, can be described as a $Transmitter - Side\ SNR\ Based\ Adaptation$. A study limited to other algorithms in this category that targeted environments which resembled our simulation environment, led to $BanditLink$ [2]. This section briefly describes this algorithm and its implementation in our simulation environment. Experimental results and evaluation are discussed in subsequent sections.

A. Description

The $BanditLink$ algorithm aims to implement a fully automatic and dynamic rate adaptation on the 802.11ac(n) link layer, parameterized by the channel bandwidth (cb), guard interval (g), level of frame aggregation (f) and modulation and coding scheme (m). This is achieved with an $exploration - exploitation$ based $automotive\ learning$ scheme defined by the $Multi - Armed\ Bandit$ system model. This system has an $agent$ which runs in two phases: (1) $exploration$, which is an initial, unguided phase that randomly explores the set of all possible link configurations (called $arms$) and measures performance ($rewards$) in each case and (2) $exploitation$, in which past experience is used to identify the best possible configuration set, given a particular channel state (defined by SNR).

In the initial exploration phase, for each set of the K possible configurations (given by $< cb, g, f, m >$), the K-armed bandit problem performs n reward measurements (called $plays$) at various values of SNR, storing the resulting set snr, k, r in a $RewardMatrix$. The subsequent $plays$ in the exploitation phase are done by probing the channel for the SNR and choosing a configuration set from the reward matrix based on an $\epsilon - greedy$ algorithm. This algorithm is designed to choose either the configuration K corresponding to closest SNR value (if one exists) or the one with the best reward overall. For each such play in the exploration phase, a new entry is added to the reward matrix. The configuration changes are done periodically, by first probing the channel for the signal strength and then determining an appropriate channel configuration with which to send subsequent packets.

The reward for each measurement is given by the Frame Success Ratio (FSR). This measure accounts for changes in the SNR, possible collisions or hidden stations interfering with communication. The data rate, however, has no direct consequence on the algorithm and its choice of channel configurations. As a result, one observation was that as long as the highest possible reward was obtained, no efforts to improve data rates were made. With the correct choice of algorithmic constants and transmission periods, this algorithm is intuitively dynamic and adaptive.

B. Implementation: Assumptions and Modifications

In order to evaluate the performance of this algorithm in our simulation environment, certain constrains and modifications have to be imposed on the K-armed bandit system model. First, the simulation environment is a singer user, single transmitter environment where the physical link layer characteristics of the channel are fixed, per iteration of the experiment. The channel bandwidth is provided as an input parameter. For the purposes of simplicity, the guard interval is left at the default value (800ns) and the A-MPDU frame aggregation value is left constant, by fixing the value of $APEPLength$ to 4096 (as in the example). As a result, the configuration set is composed solely of the MCS values, resulting in a $10 - armed\ bandit\ model$ for the 802.11ac standard.

The algorithmic constants were determined with minimum experimentation, using the input configuration of 40MHz, delay profile 'Model-D' and a distance of 4 meters. Given the nature of the simulation environment, time units are simply replaced by packet count. The duration of the initial exploration phase spans 10% of the total packet count, i.e 10 packets for a total of 100 packets. Each packet transmission constitutes an exploration period, thereby resulting in the change (or re-evaluation) of the configuration set for every packet.

The algorithmic constants c and δ do not come with selection guidelines and were chosen after limited experimentation. On running the algorithm with all combinations of bandwidth and delay profiles and an equally distributed set of 5 distance measures within the breakpoint distance (corresponding to the delay profile in each case), and iterating over a δ value between 0 and 1, with a step size of 0.05, it was seen that the performance fluctuated with no visible

trend and no consistency across configurations. A δ of 0.65 and c of 1 (for the $\epsilon - greedy\ algorithm$) were chosen at random as they gave the best results in at least some of the tested configurations.

V. Simulation results

During the final test the proposed algorithm will need to perform well on a channel with as of yet unknown parameters. As such, the algorithms are tested for many different situations. In this section results are shown for all combinations of algorithm, bandwidth and delay model at distances of one half and one times their breakpoint distance. The data in this section is based on simulations of 2000 packets each.

A. Throughput

The main performance criteria for a rate control algorithm is the data throughput it enables over a given channel. All four algorithms were simulated under the same conditions, the results of which are shown in Figure 1 and Figure 2 at a distance of one half and one times the breakpoint distance of the simulated delay profiles. In these graphs the blue, red, yellow and purple bars represent the original, averaging, weighted averaging and bandit algorithm respectively. The trend in all the algorithms is that certain combinations, such as $CBW20$ with $Model - A$ and $Model - D$ give better results than other, such as $CBW160$ with $Model - F$. The reason for this is not entirely clear however as each incremental model describes a more difficult channel with more delays and reflections. A clearer decreasing trend in performance would have been expected.

From the graphs it is clear that the performance of the first three algorithms lie very close together. There is no definite case where one algorithm is always better than the others. When averaging the overall performance of all algorithms over all delay profiles as in Table I this becomes even more clear. At one half times the breakpoint distance the weighted average algorithm appears to work slightly better than the others, but this difference in very slim.

The $BanditLink$ algorithm appears to perform worse than the other three in most cases. With a channel bandwidth of 20MHz however, this algorithm actually performs (marginally) better than the others for delay profiles of Models C, E and F at half the breakpoint distance and additionally with Model-E at the breakpoint distance. In all other cases it performs (at least slightly) worse than the other algorithms. It can be argued that the algorithm is designed for an environment with more variables that can be tuned dynamically than in our simulation environment. In addition, the algorithmic parameter choice could also not be optimal.

B. Package Error Rate

The Packet Error Rate (PER) is another important performance metric that reflects the ability of the algorithm to cope with factors such as noise, interference, synchronization issues, multi-path fading and collisions, among others. In

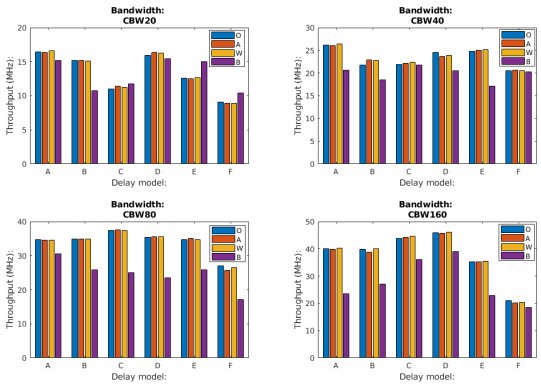

Fig. 1. Average throughput for all algorithms at 0.5 times the breakpoint distance

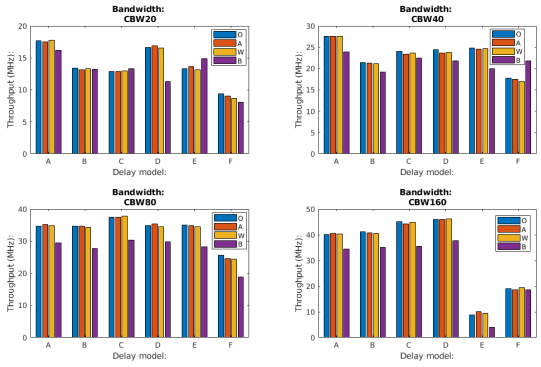

Fig. 2. Average throughput for all algorithms at exactly the breakpoint distance

the given simulation environment, simulated noise and synchronization issues are the two main causes of transmission errors. The evaluation of the algorithms in terms of the PER were done alongside the data rate experiments, with the same input parameter combinations and test conditions. The results of the simulation ran at a distance of one half and one times the breakpoint distance of the simulated delay profiles are shown in Figure 3 and Figure 4 respectively.

The first thing that pops out from the results is that the $BanditLink$ algorithm has the worst error rate of the lot, by a large margin. The main difference in the behavior of these algorithms is that the changes in MCS values for $BanditLink$ do not have a step-size limitation, i.e. they do not change to the nearest possible value, but rather to the best possible value based on the entire transmission history. In theory, this was expected to give a better result than the limited transitions in the other algorithms, but this is not the case in practice for our simulation settings.

The original, the moving average and the weighted average versions show comparable results with all input combinations, following the same trend as the measured data rates. No one algorithm (among these three) has a consistently better performance. Looking at the general trend, the worst PER measures result from $Model - F$ in all cases, except

TABLE I

AVERAGE THROUGHPUT OVER ALL DELAY MODELS

Dist:	BW:	Original:	Average:	Weighted:	Bandit:
0.5bp	20MHz	13.38Mbs	13.45Mbs	13.48Mbs	13.10Mbs
	40MHz	23.31Mbs	23.42Mbs	23.55Mbs	19.80Mbs
	80MHz	34.05Mbs	33.88Mbs	33.95Mbs	24.67Mbs
	160MHz	37.68Mbs	37.31Mbs	37.84Mbs	20.94Mbs
1bp	20MHz	13.87Mbs	13.85Mbs	13.73Mbs	12.84Mbs
	40MHz	23.31Mbs	23.00Mbs	22.93Mbs	21.52Mbs
	80MHz	33.75Mbs	33.67Mbs	33.40Mbs	27.37Mbs
	160MHz	33.33Mbs	33.35Mbs	33.47Mbs	23.84Mbs

for $CBW160$, where $Model - E$ also fares poorly. The best results for the different bandwidths are obtained with different delay profiles. $CBW20$ performs best with $Model - A$, $CBW40$ with $Model - E$, $CBW80$ with $Model - C$ and $CBW160$ with $Model - D$. This result holds true at both measured distances.

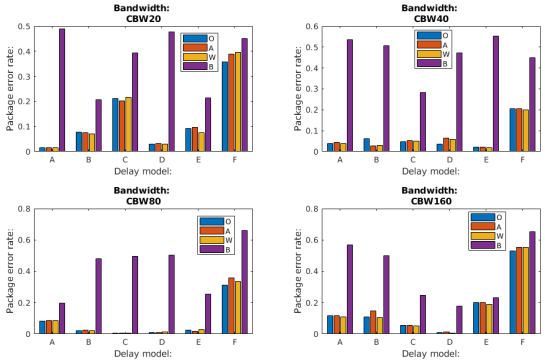

Fig. 3. PER for all algorithms at 0.5 times the breakpoint distance

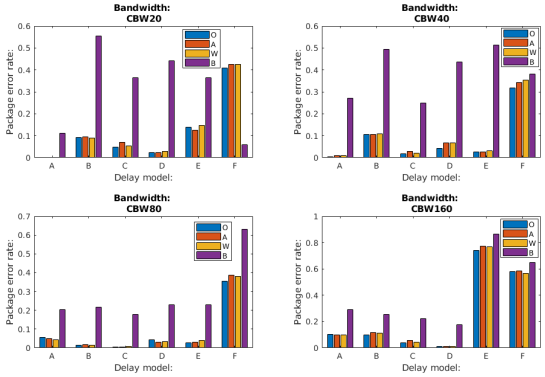

Fig. 4. PER for all algorithms at exactly the breakpoint distance

VI. DISCUSSION

During this project several problems emerged that could only be solved by time intensive simulations. As such some assumptions were made as covered in this section.

Unfortunately not all combinations of input parameters could be ran for a large amount of packets, as running one single simulation of 2000 packets at a bandwidth of 160MHz takes roughly an hour to 90 minutes on the Intel $i3 - 4350$ processor in the available desktop. Even when running multiple instances of Matlab in parallel, each running a different algorithm to fully make use of all available processor cores, simulating all possible settings would take many days. As such, the simulations were ran for all combinations of bandwidths and delay models, but for only three distances each. These were chosen to be 0.5, 1 and 1.5 times the breakpoint distance of each delay profile model. After the simulation finished after roughly three days it was found that all results at a distance of 1.5 times the breakpoint distance were identical to those exactly at the breakpoint distance. Therefore the results section only covers these two distances. Performance may have varied more for different distances within the breakpoint distance.

Further, not all possible history lengths or weights could be simulated for their respective algorithms due to the same time constraints. After running simulations consisting of 100 packets for several sizes and weights it was decided to set the history length to 6 packets, and use a weight vector of $(1, 1, 2, 2, 3, 5)$ as these seemed to lead to the best results. More thorough simulations would be needed to find the optimal settings for these variables for each channel type. The same holds true for the bandit algorithm of which not all parameters could be optimized.

VII. CONCLUSION

Several rate control algorithms were developed in Matlab and simulated under the same testing conditions. It was found that the performance of the given original algorithm, the adapted averaging algorithm and the weighted averaging algorithm is nearly identical for most channel situations. The weighted averaging algorithm performed slightly better than the others at the shorter distance, but this improvement was minimal.

The bandit algorithm based on a short literature study performed significantly worse under most circumstances, which is most likely caused by non-optimal tuning of its parameters. The same could be said for the other averaging algorithms. In both cases more simulations would be required to find optimal parameters, which would be very time consuming. This does however leave this project open for further research and improvements.

REFERENCES

[1] S. Biaz and S. Wu. Rate adaptation algorithms for ieee 802.11 networks: A survey and comparison. In *2008 IEEE Symposium on Computers and Communications*, pages 130–136, July 2008.

[2] R. Karmakar, S. Chattopadhyay, and S. Chakraborty. Dynamic link adaptation in ieee 802.11ac: A distributed learning based approach. In *2016 IEEE 41st Conference on Local Computer Networks (LCN)*, pages 87–94, Nov 2016.