IEEE 802.11n Rate Adaptation in Ath9k Driver: Implementation of Novel Algorithms and Comparison with Minstrel Algorithm

Angelis Marios
mangelis@e-ce.uth.gr
mariosag14@gmail.com

Chatzistefanidis Ilias
ichatsist@e-ce.uth.gr
iliasece16@gmail.com

I. INTRODUCTION

Nowadays, the performance of Wireless Local Area Networks (WLANs) has significantly increased by implementing novel technologies that provide multiple and high rate capabilities. IEEE 802.11n builds on previous 802.11 standards by adding multiple-input multiple-output (MIMO) and 40 MHz channels to the physical layer, and frame aggregation to the MAC layer.

MIMO is a method for multiplying the capacity of a radio link using multiple transmission and receiving antennas to exploit multipath propagation. In this way, the 802.11n devices based on OFDM technology, are able to simultaneously transmit and/or receive data through multiple antennas. Data signals are split into multiple parts. Each independent signal, called a spatial stream, is sent from a transmitter antenna. The current standard allows for a maximum of four spatial streams.

Channel-bonding/40MHz operation simultaneously uses two separate channels to transmit data, thus doubling the rate in principle. The legacy 802.11a/b/g systems use a single 20MHz channel, but 802.11n can operate in the 40MHz mode over two adjacent channels, one as the control and the other as the extension.

Frame aggregation amortizes protocol overhead over multiple frames. It packs several data frames, called Mac Protocol Data Units (MPDUs), into an aggregate frame (called A-MPDU)

A fundamental problem is that an 802.11 wireless channel suffers from time-varying losses due to mobility, interference and contention from hidden stations, leading to poor and inconsistent throughput performance. In particular, the technique of Rate Adaptation (RA) is a fundamental resource management issue for IEEE 802.11 devices; its goal is to optimize the wireless link throughput in various environments. In addition, all of the IEEE 802.11 standards do not specify any algorithm for automatic rate control. The basic idea of rate control is to estimate the current wireless channel condition and dynamically select the best data rate out of multiple available transmission rates. The wide variety of data rates provided in 802.11n high speed WLANs demands an efficient control scheme for optimal rate selection and also fast rate adaptation based on time-varying channel conditions.

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

Our work is focused on implementing novel algorithms for Rate Adaptation based on Minstrel by modifying some of his functionalities. In this way, we could conclude on what the Minstrel's vulnerabilities are and whether we could achieve an optimization. The rest of the paper is organized as follows: Section II provides some related work to ours. In Section III we analyze some basic components of Minstrel. Section IV demonstrates how our Algorithms are constructed. In Section V we administer the experimentation set up. In Section VI we show our results and in Section VII we conclude.

II. RELATED WORK

- *A Rate Adaptation Algorithm for IEEE 802.11 WLANs Based on MAC-Layer Loss Differentiation* by Qixiang Pang, Victor C.M. Leung and Soung C. Liew. They propose a new auto rate algorithm called loss-differentiating-ARF (LD-ARF) that is suitable for a realistic WLAN environment where both collision losses and link error losses can coexist, and demonstrate its effectiveness through extensive performance evaluations.
- *Rate Adaptation for 802.11 Wireless Networks: Minstrel*, Paper #86, 14 pages. This paper gives a general introduction to 802.11 rate adaptation and sources of frame loss in 802.11 networks. They discuss design considerations for Minstrel and make some initial comparisons with a selection of rate adaptation algorithms, covering the three principal families of algorithms (step up/step down, SINR, and acknowledgement feedback estimation).

- *Practical Rate Adaptation for Very High Throughput WLANs* by Arafet Ben Makhlouf, Student Member, IEEE, and Mounir Hamdi, Fellow, IEEE. In this paper, they design a practical rate control algorithm for 802.11n WLANs, based on a probing system that guarantees that it is has Long-Term Stability and Short-Term Responsiveness (L3S). They then implement it in commercial devices using the actual Ath9k driver without modifications to the existing standard.

III. MINSTREL ALGORITHM

In Ath9k, Atheros Communications specifies 4 retry series (r0/c0, r1/c1, r2/c2, r3/c3) for every frame ready to be sent, each pair with a specific rate ri and its maximum transmission attempts ci. Therefore, the lost packet is transmitted at four different decreased rates (max_throughput, second max_throughput, max_probability and lowest rate) until a success or an exceeding of the retry limit. Every frame is discarded after c0+c1+c2+c3 unsuccessful transmission attempts. In addition, the Ath9k driver uses the Minstrel's sampling algorithm which sends a constant fraction (10%) of the data packets at the adjacent rates of the current rate. This procedure is called sampling and assists eventually into rate convergence.

The functions used by Minstrel in sampling are implemented inside the file *rc80211_minstrel_ht.c*. The sequence that they are called, the one inside the other, is as follows. At first, *minstrel_ht_get_rate()* is called in order to find out the rate for the next time interval. In the case of sampling, *minstrel_get_sample_rate()* is called to obtain the rate of the specific sample. This is done by computing the index of the current sample in the table of MCS indexes:

$$sample\_idx = sample\_group * 10+$$

$$sample\_table[mg \rightarrow columns][mg \rightarrow index]$$

In this formula, the *sample_group* is the group of MCSes in which we are going to sample. Moreover, the *sample_table* is a two-dimensional matrix and each dimension has length equal to *MCS_GROUP_RATES* (10 in our case). Specifically, in every line it contains a random sequence of numbers 0-9. Each element of *sample_table* indicates one sample. The *mg* $\rightarrow$ *columns* represents the current line of the *sample_table* and the *mg* $\rightarrow$ *index* represents the current column. It is important to note that each group maintains a copy of the variables *mg* $\rightarrow$ *columns* and *mg* $\rightarrow$ *index*.

After finding *sample_idx*, *minstrel_set_next_sample_idx()* function is called in order to, finally, obtain the appropriate rate for the next sample. At first, Minstrel selects the next *sample_group* using a circular pattern. Then, the next column of this sample line is selected as the next *mg* $\rightarrow$ *index*. If *mg* $\rightarrow$ *index* has reached the end of this sample line, Minstrel selects the first index of the next line, by setting the *mg* $\rightarrow$ *index* to 0 and by increasing the *mg* $\rightarrow$ *columns* by one. Thus, it samples in all groups in a rapid way. When a sample is sent, statistics are updated. In particular, EWMA[1] probability, rate attempts, successes and throughput are kept for every MCS rate. In this way, max_throughput, second max_throughput and max_probability rate are computed in an optimal way.

IV. OUR ALGORITHMS

We developed two novel algorithms based on Minstrel, each one with different functionalities.

A. First algorithm

Like the Minstrel, so our algorithm is sampling in all valid groups, included the group to which the max throughput rate belongs. The main difference from Minstrel is that our algorithm is sampling a *sample_table*'s line for every MCS-group, instead of a *sample_table*'s element. Specifically, when *mg* $\rightarrow$ *index* is greater than *MCS_GROUPS_RATES*, a *sample_table*'s line has been completed. In this moment, the function *get_next_sample_group* is called with parameters the group in which we sample and the group in which we transmit. This function returns, in a circular fashion, the next group in which we are going to sample. Thus, the the *sample_group* variable has been set for the next line of samples.

B. Second algorithm

As in our first algorithm, so in the second, we sample a *sample_table*'s line for every MCS-group. The key difference from our first algorithm is that in this case, we prefer not to sampling in the same group that we are transmitting. For this reason the function *get_next_sample_group()* is called in two points. In order to distinguish them we use a mode-flag, which takes the value 1 or 2. The first call is being done when we have chosen the next group based on max throughput, but this is the same as the one in which we are transmitting. So, we call the *get_next_sample_group()* function in order to change the sampling group. The second call is being done when a *sample_table*'s line has been completed. Here, we call the function in the same way as our first algorithm, achieving circular access in the groups.

V. EXPERIMENTS

A. Topology

In order to test our algorithms, we use our University's Testbed [2]. We implement a four-node[3] topology as shown in Fig.1. The topology includes 2 pairs of Access Points (AP) and Stations (STA). The direction of the traffic is from STA to the AP.

[1]Exponentially Weighted Moving Average (EWMA), is a first-order infinite impulse response filter that applies weighting factors which decrease exponentially. The weighting for each older datum decreases exponentially, never reaching zero.

[2]NITOS Future Internet Facility is an integrated facility with heterogeneous testbeds that focuses on supporting experimentation-based research in the area of wired and wireless networks. It is comprised of three different deployments, the Outdoor Testbed, the Indoor RF Isolated Testbed and the Office Testbed

[3]The term node is used to describe a PC

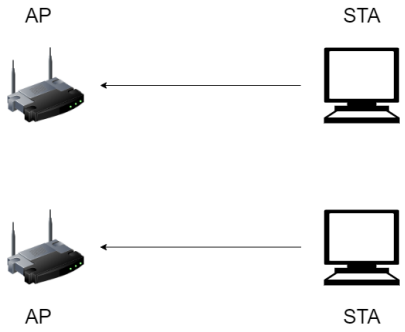

Fig. 1. Topology

B. Hardware-Software

From hardware's perspective, nodes support IEEE 802.11n wireless technology. The software we use to implement our algorithms and to evaluate Minstrel is the Ath9k backports-5.3.6-1 driver[4].

C. Execution

Our goal is to find out how fast our algorithms converge to an MCS rate in comparatively to Minstrel in difficult circumstances. Thus, we use the one pair of AP-STA to produce big interference. We adjust it to a fixed rate, in particular MCS0 rate, and we send traffic equal to 5Mbps using iperf[5].

Concerning the second pair, we observe its responsiveness in selecting the appropriate rate. We carried out a timer in order to realize when there is a rate convergence. This timer assumes that the rate selection is converged, when the MCSs with the 2 highest package success percentages, have a difference at least of 3.000 successes.

We apply this procedure and obtain measurements both for Minstrel and for our algorithms in many channel combinations. We need to observe their behavior when the interference link takes place on channel 6, where the CSMA/CA[6] link is being activated on channels 6,7,9 and 11.

VI. RESULTS

A. First Algorithm

In Fig.2 are shown the mean timer results for our first algorithm and Minstrel. It is obvious that our algorithm comes slower to convergence. This is happening, because it is wasting time at the beginning of the transmission, by sampling in all groups a *sample_table*'s line. Specifically, our algorithm initially transmits in group 0, when all statistics (EWMA probability, rate attempts, successes and

[4]Ath9k is a completely FOSS wireless driver for all Atheros IEEE 802.11n PCI/PCI-Express and AHB WLAN based chipsets.

[5]iPerf is a tool for active measurements of the maximum achievable bandwidth on IP networks. It supports tuning of various parameters related to timing, buffers and protocols

[6]CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) is a protocol for carrier transmission in 802.11 networks. Unlike CSMA/CD (Carrier Sense Multiple Access/Collision Detect) which deals with transmissions after a collision has occurred, CSMA/CA acts to prevent collisions before they happen.

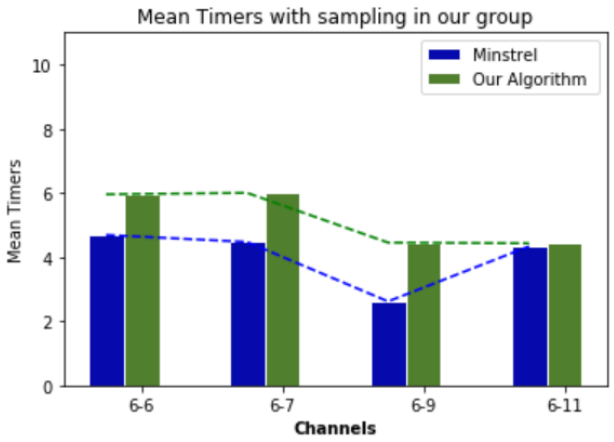

Fig. 2. Minstrel - Our Algo 1: Mean Timer

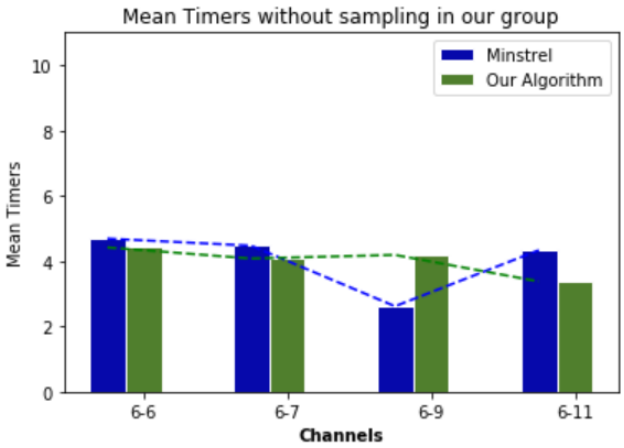

Fig. 3. Minstrel - Our Algo 2: Mean Timer

throughput) are not yet calculated. Then, it starts sampling a *sample_table*'s line in the group 0 and then serially in all the following, until it reaches the group with the highest rates. In the contrary, Minstrel is sampling a *sample_table*'s element in a circular fashion in all groups. In this way, in the same time interval the latter has sent more samples in the group with the greatest rates and thus it is more fast in throughput calculation and rate selection.

In Fig.4 are shown the mean success attempts in the top 3 highest-throughput rates between our first algorithm and Minstrel. Both implementations seem to converge in MCS-23 rate. Also, we observe that there is a similar behaviour between the two algorithms with Minstrel to slightly exceed ours in MCS-23 successes. This is happening because Minstrel comes faster in convergence.

B. Second Algorithm

In Fig.3 are shown the mean timer results for our second algorithm and Minstrel. Here we could generally observe that our algorithm comes slightly faster in convergence. This is done by skipping sampling in the same group that it is transmitting. Specifically, our algorithm initially transmits in

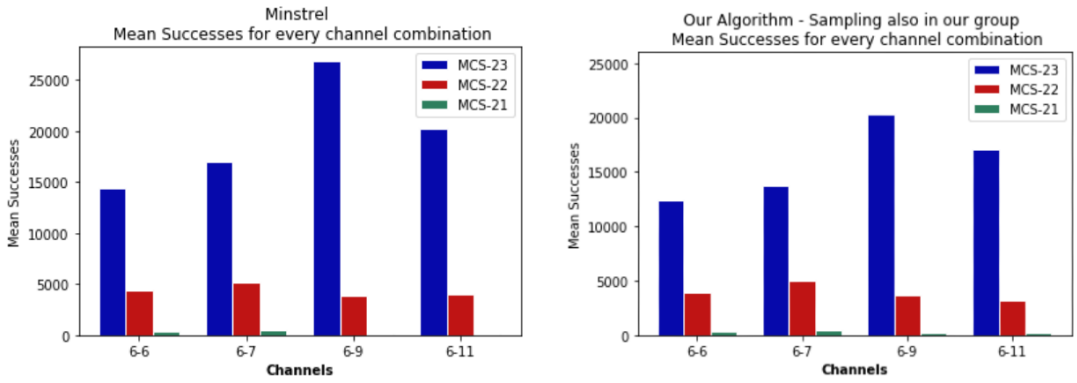

Fig. 4. Minstrel - Our Algo 1: Mean Successes in top 3 MCS rates

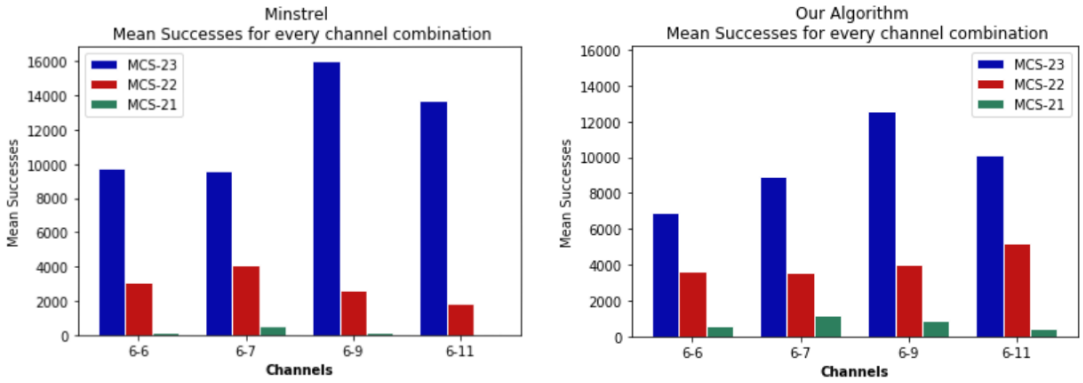

Fig. 5. Minstrel - Our Algo 2: Mean Successes in top 3 MCS rates

group 0 and then it starts sampling in group 1, skipping group 0. In addition, it is sampling a *sample_table*'s line for every group, obtaining much more statistics in order to select rate accurately.

In Fig.5 are shown the mean success attempts in the top 3 highest-throughput rates between our second algorithm and Minstrel. Both implementation seem to converge in MCS-23 rate. Moreover, our implementation seems to alternate between MCS 21, 22 and 23. This means that there are more sample packet successes in MCS 21 and 22 from Minstrel. The reason is that when we finally transmit with rate MCS-23 in group 2, we will not sample again in this group. Nevertheless, the essential difference of 3.000 successes always exists, between the 2 highest package success percentages. So, it is able, most of the time, to converge in MCS-23.

VII. CONCLUSIONS

From our survey we realized that rate adaptation is very critical for the overall performance and thus a powerful rate adaptation algorithm is necessary. Minstrel algorithm is completed and very robust in comparison with our algorithms. It is equipped with a strong MCS-access pattern, that can obtain enough statistics rapidly for all available MCSs, accessing all groups in a circular fashion. In this way, it acquires speedily an overall picture of channel condition and it comes immediately to the appropriate rate convergence.

ACKNOWLEDGMENT

This work is within the scope of the course ECE436 Wireless Communications at Department of Electrical and Computer Engineering, University of Thessaly, Volos, Greece.

REFERENCES

[1] A.B. Makhlouf and M. Hamdi, IEEE, "Practical Rate Adaptation for Very High Throughput WLANs", IEEE transactions on wireless communications, Vol. 12, No. 2, FEBRUARY 2013,908
[2] "Rate Adaptation for 802.11 Wireless Networks: Minstrel", Paper 86, 14 pages.
[3] Qixiang Pang, Victor C.M. Leung and Soung C. Liew "A Rate Adaptation Algorithm for IEEE 802.11 WLANs Based on MAC-Layer Loss Differentiation"
[4] Ioannis Pefkianakis, Yun Hu, Starsky H.Y. Wong, Hao Yang, Songwu Lu UCLA Computer Science, USTC, IBM Research, Nokia Research, "MIMO Rate Adaptation in 802.11n Wireless Networks"