Wireless Communications
Practical Rate Adaptation for Very High Throughput WLANs

Konsoulas Konstantinos
AEM: 2975
kkonsoulas@uth.gr

Ioannis Roumpos
AEM: 2980
iroumpos@uth.gr

Evangelos Zampras
AEM: 3061
ezampras@uth.gr

Evangelos Balamotis
AEM: 2927
ebalamotis@uth.gr

3 July 2023

Contents

1 Introduction

In this assignment, our objective is to implement the L3S (Long-Term Stability and Short-term Stability) rate adaptation algorithm in the ath10k driver. The L3S algorithm aims to enhance the performance and stability of wireless communication by dynamically adjusting the modulation and coding scheme (MCS) based on both long-term and short-term measurements.

By incorporating the L3S algorithm into the ath10k driver, we seek to improve the throughput, reliability, and overall quality of the wireless link in various network environments. Having said that we will be examining the current implementation of the existing minstrel algorithm and identify various areas in which the proposed algorithm aims to improve the overall performance. Finally we will be performing some experiments, extracting insight from the gathered data.

2 Minstrel Algorithm

The implementation of the minstrel algorithm in wireless communication systems involves various components and processes. It focuses on dynamically adjusting the transmission rate based on channel conditions and link quality. By continuously monitoring the performance of different rates and adapting rate selection, the algorithm aims to optimize throughput while ensuring a reliable connection.

It maintains statistics on success and failure rates, utilizing this data to estimate transmission probability and make informed decisions. The algorithm's feedback loop periodically updates rate selection to respond to changing conditions, prioritizing higher rates when quality is good and lower rates when it degrades. The available rates are maintained into a statically defined table that are based on the available Modulation and Coding Schemes (MCS) of the wireless system.

3 The Proposed Algorithm

The Long-Term **S**tability and **S**hort-term **S**tability Algorithm (**L3S**) builds upon the existing minstrel algorithm using new metrics and techniques in order to extend and optimise the existing implementation. The L3S algorithm maintains long-term and short-term statistics for each used transmission rate. These statistics help estimate the channel state and guide rate adaptation decisions. Long-term statistics (e.g., achieved throughput, packet error rate) are used for system stability, while short-term statistics handle transient variations for system responsiveness.

Additionally the proposed algorithm utilizes a multi-rate retry mechanism to handle short-term changes in link quality. It retries lost packets at different data rates, progressively decreasing the rate until success or reaching the retry limit. The dynamic nature of L3S makes it ideal to varying channel conditions, enabling the wireless interface to adapt accordingly and maximising effortlessly the throughput currently available.

3.1 Implementation Details

The modifications necessary needed to the existing minstrel algorithm driver code took place into the rc80211_minstrel_ht.c and rc80211_minstrel.h files. Inspecting carefully the rc80211_minstrel.h file we are able to identify the struct minstrel_rate_stats which is designed to store statistics and measurements for a given rate. In the aforementioned struct we opted to embed the variables *consecutive_success* , *consecutive_failures* and *consecutive_retries* which will be used by the proposed algorithm and will be updated and initialised in other parts of the code. This statistics are for the **short-term** rate adaptation mechanism.

For the **long term**, statistics are established in order to provide best throughput. The best possible rate is calculated by calculating the throughput performance that has been seen, whether it has been employed in previous transmissions or by the adjustment that improves system stability. From the transmission state to the probe state, the L3S algorithm switches. The long-term statistics are updated whenever the algorithm changes states. The algorithm is in transmission mode whenever a new round starts, and data is delivered via the previously specified MRR series.

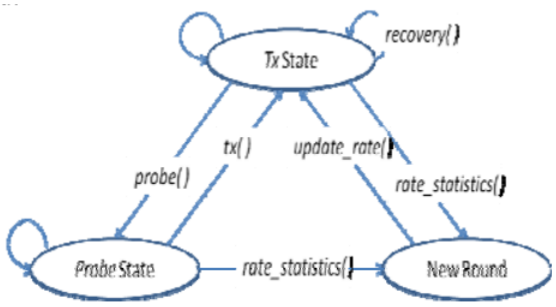

Figure 1: States of our algorithm.

In our implementation to achieve this we found the ieee80211_tx_rate struct where there are fields to keep track the index, for the rate, and a field of count where the maximum transmission attempts are stored.

The **minstrel_ht_tx_status** function is called when a transmission status report (such as an ACK) is received for a packet. In this way it is responsible for updating the internal state of the algorithm. In our implementation specifically it it tasked to collect the new L3S short term statistics mentioned above and declare a possible update for the current rates by calling the update_rates function.

The **minstrel_ht_get_rate** function is used in order to get the current rate based on the index of MCS and the number of attempts in the same rate. In our implementation, we extended this function and update the variables of rates and maximum attempts of retransmissions based on the L3S algorithm. In more detail, in our implementation depending the state, either transmission or probe state, the rate is calculated based on different MCS indexes. In the transmission state, there are used three consecutive rates in decreasing order while the current rate is used for the rix1.
In the probe state things are bit more complex. Whenever, there is a probe we count how many times the probe happened. In the first series, the probe happens at neighbouring MCS indices and the MCS group can change only when the current rate is already at the lowest MCS index. On the other hand, in the second probing series/period the probe direction happens depending the group the index lies to. In the case that index lies in the rightmost group, the index changes and goes to a a left group. In the same way, if an index is in the leftmost group it changes and goes to a right-one. The multi-retry mechanism stays the same as the number of attempts are two for probe state rates.

4 Experiments

To test the proposed rate adaptation algorithm L3S we use the University's Indoor Testbed (NITOS). We implement a 3 node topology, where nodes are either a Station or an Access Point , as shown in the following figure.

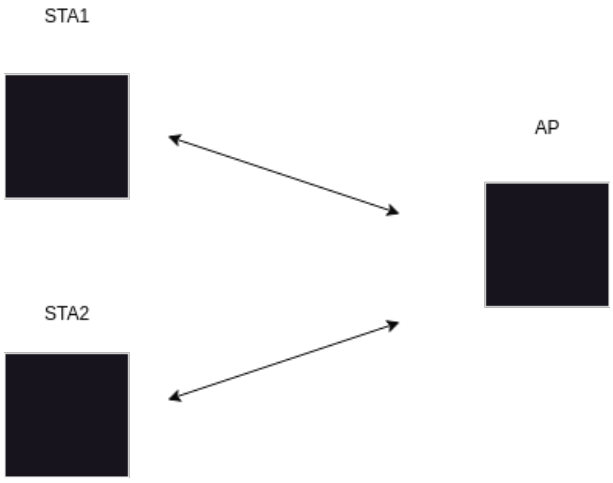

Figure 2: Topology of the experiment.

The topology includes 2 Stations and 1 Access Point. From a software perspective we used the Ath9k backports-5.3.6-1 driver and from the hardware perspective the nodes support IEEE80211n technology.

The goal is to measure the throughput rate and compare it with the original (Minstrel) rate adaptation algorithm through different scenarios. In the first scenario, in a fixed channel, we generate traffic from one station to the access point for a long period of time around 200 seconds. This station is the one with the implementation of our proposed algorithm. In addition, we connect and another station to the access point in order to create interference and see how well our proposed algorithm adjusts to the channel conditions. This second station sends data periodically for the duration of the experiment.
In the second scenario, every one minute we change statically the channel in which the experiment is conducted. In that way, we can compare how our rate adaptation algorithm is adjusting through different channel configurations.
In the first scenario, we generated traffic using **iperf** for 200 seconds with intesity of 20 Mbps and in the interference link the station node is generating traffic for a specific time interval of 20 seconds with a timeout of 15 seconds.
The statistics of this scenario where collected three times and the average throughput is computed.

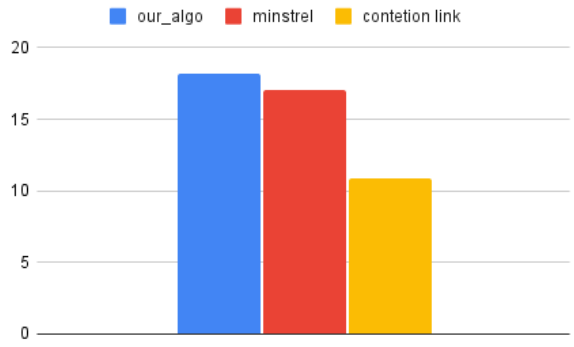

Figure 3: Throughput with interference link

As we see our proposed algorithm outperforms the minstrel one but without any significant change. This is explained because the channel conditions are not changed in a way that the algorithm handles these changes.

In the second scenario, every one minute we changed statically the channel from 3 to 10.

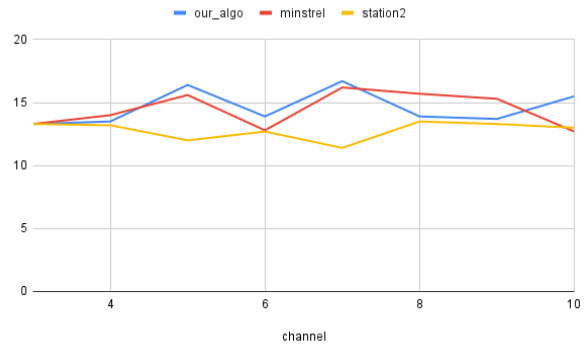

Figure 4: Throughput with channel changes.

As we see, in some channels our algorithm (L3S) outperforms the basic one but in bigger channels the minstrel one is better from the implemented one. Generally, in different channel characteristics such as frequency, capacity, interference and signal quality play a significant role in the rate adaptation algorithm. Having said that, one possible explanation of the figure above is that the implementation of our algorithm is not taking care of all possible scenarios for the changes of the channel conditions.