Cognitive WI-FI: A small step towards online self-learning data rate control

Jetse Brouwer
Delft university of technology
Embedded Systems
Mekelweg 4, 2628 CD Delft
Email: j.brouwer-3@student.tudelft.nl

Niels Hokke
Delft university of technology
Embedded Systems
Mekelweg 4, 2628 CD Delft
Email: n.h.hokke@student.tudelft.nl

***Abstract*—**

1. Introduction

With the use of higher modulation and coding schemes (MCS) higher data rates can be achieved. however, a higher MCS also results in a higher susceptibility for noise. To maintain high data rates while ensuring connectivity a trade of has to be made between speed and availability. While algorithms do exist, they are mainly static. Our goal is to create a self-learning algorithm that dynamically learns from its environment without downtime. The algorithm is designed and tested by using the 802.11 Dynamic Rate Control Simulation example from the MatLab WLAN toolbox

2. Method

The algorithm exploits the correlation between the probability of successfully decoding a packet at a given MCS and the signal-to-noise ratio (SNR). When lowering SNR the probability for successfully decoding a package at a given MCS reduces significantly. The aim of our algorithm is to learn about this correlation and use this to its advantages. If the algorithm, for every possible SNR, would know the best suitable MCS it would be able to always perform at its maximum performance. Since the correlation does not necessarily has to be linear, and might change depending on channel configuration a single pre-defined correlation will not work. To know which MCS would be the best suited for the given SNR the resulting bandwidth of the data transfer needs to be stored for later references. Besides storing the previous results it should also explore other, possibly better values for a given MCS.

2.1. Data structure

To use the correlation to our advantage, the data aggregated from previous measurements needs to be stored. This is be done by using a two-dimensional array, the 'LearnedValues' array. The first dimension corresponds to the SNR value. The SNR is divided in to bands where every x dBs are grouped together to a single index. This is calculated using the formula $index = \lfloor \frac{SNR}{x} \rfloor$. The second dimension corresponds to one of the ten available MCS indices. A single element from the array then represents the data rates that can be achieved for that specific MCS and SNR combination.

If a package is decoded successfully or not is not constant, but a probability based on the SNR and MCS. Therefore when a new value is detected it should not directly overwrite the old value but create a combination of both the old and current data rate. How fast the algorithm should adopt this new value is set with the variable 'adoption rate'. This variable is a value between 1 and 0 and corresponds to the portion of the current value in the new value. The new value is given by our value-adoption formula: $Rate_{new} = Rate_{old} \times (1 - adaptionrate) + Rate_{current} \times adaptionrate$

2.2. Learning the data rates

Based on the data rates the algorithm has learned, it can pick for a given SNR the best suitable MCS. It does so by first using the SNR from the previous packet as a starting point. From here it will look in the array at the column that matches that SNR range. It will look up the maximum know data rate it has seen, and use this as a possible candidate for the MCS. While history seems to be repeating, it does not always gives enough insight to act on in the future. Therefor the algorithm should also explore other values besides the best know value. Therefore the algorithm uses the variable 'LearningRate', which has a value betweeon 1 and 0, and is the probability of trying out a new MCS. The value of this MCS is chosen by the current ideal MCS plus 1, the result from this exploration will be stored in the 'LearnedValues' array. With this method the algorithm is always looking for better options.

If one of the known values becomes less suitable for some reason, this will be reflected in the 'LearnedValues' array. Based on the adoption rate it may take one or more failed packets to not be the maximum data-rate for the given MCS SNR combination anymore.

2.3. offline vs online learning

Based on how many packets are being send and the learning rate, the speed by which the algorithm learns can vary greatly. However with Wi-Fi supporting up to 3000 packets per second [1] it is expected that the algorithm will quickly adjust. One of the decisions that has to be made is the value of the learning rate, a high learning rate will result in a quickly adjusting system. However, when the system is reaching it's maximum it will still try to increase the MCS proportional to the learning rate. If the learning rate is set to 10% and is at it's maximum, up to 10% of the packets will be lost and therefore reduce the throughput

To speedup the learning process of the algorithm an offline training mechanism is implemented. This ignores the current maximum value and sends out as much packets as possible spread out over all possible MCS to gain knowledge about the correlation between the current SNR and MCS. In the simulation this is implemented by running the 100 packet long pattern for all 10 indices, 10 times each, resulting in 10000 packets to be send.

3. Results

After performing an offline learning run with an adoption rate of $\frac{1}{3}$ and a bin size of 1 dB we reached an overall data rate of 26.0572 mpbs. this is an 25% improvement over the 23.295Mbps of the original. To measure the performance of the online learning algorithm multiple consecutive runs have been done. The chosen learning rates for the test were 1%, 5% and 10%. To have a fair start they all started with a run without having learned anything, from there on the algorithm was free to learn as it desired.

From picture 1 it can be seen that the 10% indeed learns the fastest, but also is very unstable as it is very aggressive with trying out new values. With 5% the algorithm learns a bit slower, but still reaches the peak quiet fast. Compared to the other the 1% learning rate learns significantly slower. Over time all three settings seem to converge to the same value. As it only took 20 runs to reach the same value as the others, but is more stable it seems to be a better value when it comes to user experience.

4. Conclusion and further work

With an overall data rate of 26 mpbs the dynamic algorithm described in the paper shows an improvement of 25% over the original one. As the algorithm is self learning it will be able to adapt to its environment.

As the system begins with the lowest MCS the chance of finding an better MCS is way higher at the start of the run than at the end. Later in the run the chance of finding a to high MCS which results in a failed package becomes larger and larger. The current learning rate is a static number, every x samples a new MCS is tested. It would be better to start with a high learning rate and slowly reduce it further in the run. An dynamic learning rate could result in faster learning while reducing the number of failuers due to high MCS values. further research can be done on how to dynamically change this learning rate for the optimal result.

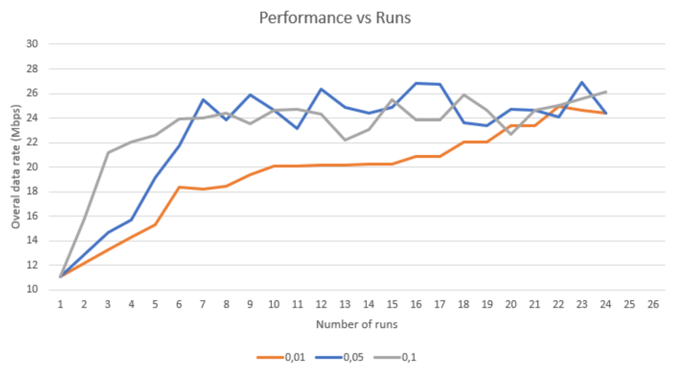

Figure 1. Performance over runs (100 packets per run) for different learning rates

References

[1] Pengyu Zhang, *Enabling Practical Backscatter Communication for On-body Sensors*, SIGCOMM 16, August 22-26, 2016, Florianopolis , Brazil