Wireless IoT and LAN: Rate Control Algorithm Report

Jiaxuan Zhang (5258162, jiaxuan zhang) Yuan Fu (5215315, yuanfu)

March 30, 2021

Report contains 4 of maximum 4 pages.

1 Basic Idea for the Rate Control Algorithm

In wireless networks, the channel conditions are unpredictable, leading to varying transmission quality. The transmission rate should be controlled dynamically since good channel conditions pave the way for high throughput and low error rate, whereas poor channel conditions should limit the transmission rate to reduce additional re-transmissions. There are some rate control algorithms that inspire us. From[1] we know that in Minstrel method, the data rate is based on the percentage of successful transmission, which is updated for an interval of a defined time using Weighted Moving Average(WMA). It also uses a portion of the packet to probe for an optimal rate. This method inspires us to use WMA and probe for higher throughput when channel conditions are good.

We have considered three ways of rate control algorithm: Control-Based Algorithm, Reinforcement-Learning Based Algorithm, Machine-Learning Based Algorithm.

The Control-Based algorithm mainly has three specifications.

1. Instead of using a fixed threshold, a feedback loop adjusts SNR-MCS mapping threshold dynamically based on estimated SNR, Packet error rate (PER), and Bite error rate (BER).
2. A compensation module is used to directly change MCS value based on a short-term history of BER and PER. The compensation value (we call it bias) happens more frequently than the SNR-MCS mapping threshold update.
3. Instead of change MCS at most 1 each time, we use direct mapping from MCS and threshold interval

The first specification's intuition is that a long-period small PER and BER always imply the current SNR-MCS table is conservative. It is mainly based on long-term statistics and may not be sensitive to short-term signal characteristics. The second specification focuses more on signal's short-term trend. It will directly change the MCS selection without changing the mapping table. The detailed structure and parameter settings of the control-based algorithm are in Section 2.

Reinforcement Learning-Based Algorithm assumes an agent selects MCS based on PER, BER, SNR, and previous MCS selections. However, we discard it because of two main drawbacks:

1. The state space is too large if the agent need sufficient information
2. converged learning process always needs a discount rate $\gamma < 1$ in its value function, but different time steps should be of equal importance in the entire communication process.

. For the Machine-Learning based algorithm, we predict the best MCS choice via a trained model. The main procedures are as follows:

1. Sample collection. We collect approximately 700 data which varies in SNR and delay model as the sample. Based on a utility function, each element of the sample corresponds to the best MCS choice. The list of MCS choices is the target.
2. Data cleaning. We impute data whose SNR value is NaN with 0, then encode the target for further prediction.
3. Model training. We feed the sample and target into a neural network and obtain a trained model, which is able to output the best MCS of the highest possibility.
4. Compensation. Implement some compensation in case of misprediction and to improve the performance.

. According to the above explanation, we will focus on the control-based algorithm and machine-learning-based algorithm in the following sections.

2 Implementation of the Rate Control Algorithm

In this section, the implementations of the Control-Based Rate Control Algorithm and the Machine-Learning-Based Rate Control Algorithm will be presented in detail.

2.1 Control-Based Algorithm

For the control-based algorithm, we assume an instantaneous feedback channel. The architecture of the control-based algorithm is shown in Figure 1.

The system will first predict the next-step SNR and then use the SNR-MCS table to select an MCS l. Then a compensate module will change the MCS selection to l', l' will then be used as the next-time step MCS. The system collects the BER and PER, and two loops change the the SNR-MCS table and compensation module dynamically.

For SNR prediction, we tried two methods: weighted moving average or using current SNR directly. They can be formally described as equation 1. After testing, the weighted moving average method always has

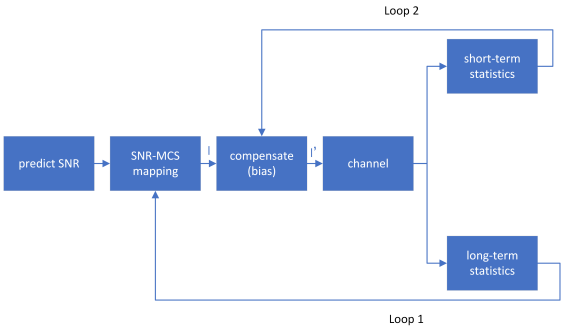


Figure 1: control-based RCA architecture

better accuracy when `meanSNR` and `maxJump` is relatively small, which represents a more smooth SNR wave. Considering it cannot be promised in each run, we finally decide to use the current SNR directly as next time SNR.

$$\begin{aligned} \mathrm{SNR}_{t+1} &= 0.89\mathrm{SNR}_t + 0.1\mathrm{SNR}_{t-1} + 0.1^2\mathrm{SNR}_{t-2} \\ \mathrm{SNR}_{t+1} &= \mathrm{SNR}_t \end{aligned} \quad (1)$$

The SNR-MCS mapping module directly selects an MCS value based on the SNR-MCS table. We use the same MCS threshold as the template, which is $[11, 14, 19, 20, 25, 28, 30, 31, 31]$. When mapping, the MCS will be i if the SNR locates in the ith interval. The SNR-MCS mapping table is updated every 30 packets by Loop1.

The compensation module is used to add bias on the MCS selection. When the current threshold is more radical than the initial threshold, the bias can only be non-positive. Otherwise, it can be 0, positive or negative. The compensate value is updated every 5 packets by Loop2.

Loop 1 is the long-term feedback loop. It will adjust the SNR-MCS table every 30 packets (an interval). It holds two variables `current_channel_ber` and `current_channel_per`. The first variable record the average BER per packet in the interval, while the second record the number of packets lost. After each interval, a `compensate` function is used. In the `compensate` function, if the `current_channel_ber` or `current_channel_per` is small, which means the current channel has low BER or small PER, each component in the table will decrease 1, vice versa. This loop adjusts the MCS selection to be more radical or conservative by updating the table.

Loop 2 is a short-term feedback loop. It updates the bias every 5 packets (an interval). The feedback controller holds two variables `current_signal_per` for PER in the interval and `current_signal_ber` for BER in the interval. The update bias is based on the rule shown in 9. One special setting is the bias can only be positive when the current SNR-MCS table is more radical than the initial table, while it can always be negative,

The maximal change on the SNR-MCS table's value is 4 because 4 is sufficient to realize a two-level MCS difference. For example, for SNR=13, if the table is decreased by 4, the MCS selection will be 0, while it be 0

Algorithm 1 bias updating algorithm

Input: *bias*: current bias; *ber*: BER in the current interval; *per*: PER in the current interval; *threshold*: initial SNR-MCS table; *table*: current SNR-MCS table; *window*: size of current update step

Output: *bias*

```
if ber=0 or per ≤ 0.1 window then
    if bias < 0 then bias=bias+1;
    else if bias<2 and table-threshold >0 then bias=bias+1;
    else if bias<2 and table=threshold then bias=bias+1;
    end if
else if ber > 0.01 or per > 0.2 window then
    if bias > -2 then bias=bias-1
    end if
end if
```

Figure 2: Confusion Matrix

if the table is increased by 4. We also set the compensation should be in the range $[-2, 2]$, which guarantees the bias can offset most of the effect of dynamically SNR-MCS table change.

2.2 Machine-Learning Algorithm

First, we use a utility function that can select the most appropriate MCS value given a configuration set. The utility function is :

$$U = data\_rate^2 + 1000 \times (1 - error\_rate) \quad (2)$$

We want to maximize the data rate while keeping the error rate at a low level. The MCS value, which causes a higher data rate but a very low error rate will be discarded. The value with the highest utility will be chosen as the label of the corresponding configuration set. To make an ML model, we need to consider the features that are relevant to the predicted value. The original sample has 6 features: delay model, mean SNR, amplitude, maxjump, estimated SNR, and the distance between the receiver and the transmitter. Obviously, mean SNR, amplitude, and maxjump have a compound influence on the estimated SNR. After applying PCA(Principal component analysis) to the original data sample, we find that the estimated SNR and the delay model play the most important role in the prediction. We then impute the NaN value of SNR with 0, and encode the MCS value using one-hot, which encodes different value of a data a unique value. For example, MCS value of 1 is encoded into

[0,0,0,0,0,0,0,0,1,0]. Following the procedures above, we start to train a neural network. The training is repeated while changing the number of hidden layers and the initial values, and we choose the model with the lowest test error, which is about 20%. One of the important features of ML algorithms is imperfect prediction. To compensate for this, we first use the WMA to get an SNR value for the input, which is shown as follows:

$$I_i = \frac{1}{6}\mathrm{SNR}_{i-2} + \frac{2}{6}\mathrm{SNR}_{i-1} + \frac{3}{6}\mathrm{SNR}_i \qquad (3)$$

, where M is the list of measured SNR and i the current packet number. For the first two packets, we just use the average. The second compensation is to increase(decrease) the MCS every defined interval if the transmission condition is good(bad). We use the packet error to quantify the transmission condition. If the number of packet errors within the interval is larger than a threshold, the compensation increases, and vice versa. Figure 2 is the confusion matrix of the test set, where we can see most of the prediction lay between [target-1,target+1],thus we set the compensation value to be -1 or 1.

3 Results

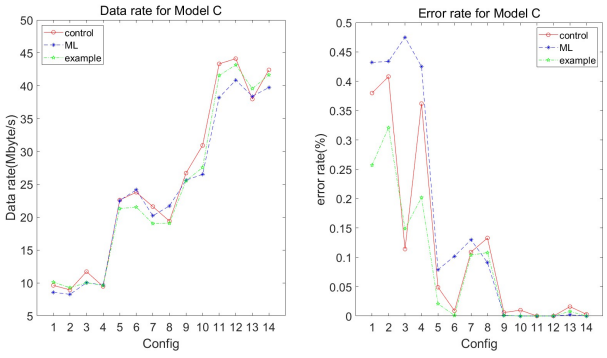
Follows are the test results of our algorithms. We set the configuration of channel conditions from bad to good to test the performance in different scenarios. We present the detailed result for Model A in Table 1. For Model C(Figure 3) and Model E(Figure 4), we only present the plots.

Config	DR_cl	PER_cl	DR_ML	PER_ML	DR_ex	PER_ex
1	11.04	0.25	14.21	0.22	8.86	0.30
2	4.44	0.65	4.96	0.62	4.42	0.62
3	13.03	0.04	15.02	0.09	10.00	0.13
4	4.96	0.60	4.77	0.63	4.98	0.56
5	31.64	0.00	31.47	0.01	23.07	0.00
6	28.16	0.00	28.34	0.01	19.09	0.00
7	21.77	0.07	27.06	0.10	23.11	0.03
8	21.82	0.06	21.34	0.15	19.46	0.03
9	35.51	0.00	32.01	0.00	30.52	0.00
10	30.74	0.00	28.38	0.00	22.34	0.00
11	44.11	0.00	43.65	0.00	43.22	0.00
12	42.80	0.00	41.55	0.00	41.25	0.00
13	43.03	0.00	42.22	0.02	41.24	0.00
14	39.13	0.01	40.50	0.02	38.32	0.00

Table 1: Result for Model A

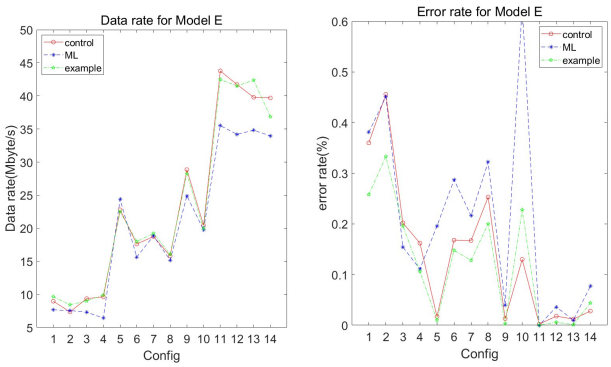
Config	1	2	3	4	5	6	7
meanSNR	10	10	10	10	22	22	22
amplitude	7	7	3	3	14	14	14
maxjump	3	3	0.5	0.5	0.5	0.5	5
distance	1	50	1	50	1	50	1
Config	8	9	10	11	12	13	14
meanSNR	22	22	22	40	40	40	40
amplitude	14	3	3	14	14	14	14
maxjump	5	0.5	0.5	0.5	0.5	5	5
distance	50	1	50	1	50	1	50

Table 2: Configurations



(a) (b)

Figure 3: Result for Model C



(a) (b)

Figure 4: Result for Model E

3.1 Control-Based Algorithm

From Table 1, the control-based algorithm performs higher data rates with slight changes (tiny increase, most equal, and sometimes decrease) on PER comparing to the template methods on Model A in most cases. Sometimes, the control-based algorithm has a lower data rate than Machine-Learning based algorithm. For Model C and Model E, the control-based algorithm perform worse than Model A, with about 2/3 and 1/2 cases better than the example template.

The control-based algorithm is more likely to beat the template method when `meanSNR` is high, with both higher data rate and lower PER. One phenomenon in Model C and Model E is that the control-based algorithm always beat the template method in 4,5,9,11,12,14 while losing 1,2,13. It may be due to adjustment lagging or fault estimation property of this method, which will be discussed in detail in the next section.

3.2 Machine-Learning Algorithm

The overall performance of the ML algorithm is better than the example for Model A. In the low SNR scenario, our ML algorithm has higher data rates and relatively lower PERs. In other cases, the data rate is still high, with the error rate at the same level. However, when the maximum jump is high, the possibility of bit error increases. This is because the large SNR change jump makes the input SNR for prediction inaccurate. The compensation is hard to adjust, causing

the MCS prediction and the compensation to be stagnant. This situation can be mitigated by lowering the update window. Nevertheless, this renders the algorithm less stable in small jump scenarios.

However, the ML algorithm is unstable and performs worse than the example in other delay models. We can see from Figure 3 and 4 while in Model C and Model E, the algorithm has the same level of data throughput with a larger possibility of bit error. The main cause is probably the biased training sample or the missing important input features. Further limitations will be discussed in Section 4.

4 Limitations of our Approach

4.1 Control-Based Algorithm

A control-based algorithm can always obtain a good solution. However, it has some significant drawbacks.

- Parameter sensitive: A control-based algorithm is always parameter sensitive. In order to get an optimal solution, lots of parameter turning work is needed.Besides, parameter settings optimal for a given test group may not beneficial another group of test. It is always hard to find a really optimal solution.
- Adjustment Lagging: The modification on parameters from loop1 and loop2 is lagging, because it always based on history not on prediction. If channel environment change dramatically, the lagging adjustment or compensation may deteriorate the performance because the bias is not appropriate for future.
- Fault Estimation: Randomness always exists in the channel, the loops, especially loop2 may wrongly estimate the trend of signal/channel. The feedback mechanism may mistake randomness in channel condition variation to essential change, which deteriorate the performance.

Limitation 2 and 3 are potential explanation for poor performance in test 1,2,13.The ratio of `amplitude` to `meanSNR` and `maxjump` to `SNR` in these groups are relatively larger than other groups. It implies in the channel, the effect of randomness may be more significant and the fluctuation is more dramatically.

4.2 Machine-Learning Based Algorithm

There are some limitations to our ML algorithm originating from the ML method itself or the collected data sample.

- Misprediction. Our ML algorithm has an 80% accuracy. As in 2 even though the predictions lay mostly in the range of [target-1,target+1], there are still some outliers. The mischoice of the best MCS increases the bit errors and lowers the data rate. In addition, our utility function is calculated based on the data rate and error rate over a range of periods. The averaged value cannot represent the instantaneous value. Consequentially, our algorithm may be biased, and the prediction may not be the best MCS of the current environment.
- Overfitting and underfitting. Overfitting and underfitting are rooted in the training data and the choice of input channel features. On the one hand, even if we select the SNR and channel properties randomly when obtaining the data, there is still a chance the data is biased if the sample number is not large enough. The predictor is overfitted and can predict more accurately in a specific delay model if the sample size is biased toward the model. On the other hand, we only make predictions based on the estimated SNR and the delay model, which may be the source of underfitting. If we can find more relevant features to the target value, this problem can be mitigated.
- Drastic change of the SNR environment. As discussed in Section 3, the ML algorithm has larger errors in large SNR jump situations. This is caused by our compensation mechanism. Even though our prediction is instantaneous, the compensation is stagnant and is updated only after a defined window time. This is the trade-off we have to make to determine the algorithm's adaptivity in either a fast change or a slow change environment.
- Offline training. The ML model is trained before we put it to real practice. However, this method limits the generality and may cause the algorithm less accurate in foreign regions. One solution to this is online batch training. Namely, we collect data when the model is working and retrain the model after a certain period. However, this also has some hardware and complexity overheads.

5 Conclusions

In this article, we detailedly discussed control-based and machine-learning-based rate control algorithms and used different test groups to test their performance (with Model-A, Model-C, and Model-E). The control-based algorithm uses two loops to adjust the threshold and compensate MCS selection dynamically. The machine-learning-based algorithm takes the delay model and estimated SNR as input and predicts an optimal MCS selection. Both of these algorithms perform better than the template example in Model A, while the machine-learning-based one is more unstable when changing to Model C and Model E. Finally, we presented the limitations of these two approaches in Section 4.

References

[1] W. Yin, P. Hu, J. Indulska, M. Portmann, and Y. Mao, "Mac-layer rate control for 802.11 networks: a survey," *Wireless Networks*, vol. 26, 07 2020.