ET4394 Wireless Networking

Algorithm for Dynamic Transmit Rate Control

Group Name: JMDB

Anup Bhattacharjee	4738845	A.K.Bhattacharjee@student.tudelft.nl
Rahul Vyas	4747798	r.r.vyas@student.tudelft.nl

TU Delft

Contents

1. Objectives

The main aim of this project is to create a new rate control algorithm for IEEE 802.11 (automatic selection of rate and coding scheme). The algorithm is evaluated in terms of Throughput and Bit Error Rate(BER), when compared to the original algorithm of the TransmitRateControlExample.m simulation file.

2. Theory and Original Algorithm

Before going forward to explain the algorithm used, lets first see the points to keep in mind while constructing the algorithm.

The algorithm in consideration changes the Modulation and Coding Scheme(MCS) of the current packet to get an improved Throughput and minimized BER. Corresponding to each MCS value there is a specific modulation Scheme and a coding technique.

MCS	Modulation	Coding Rate
0	BPSK	1/2
1	QPSK	1/2
2	QPSK	3/4
3	16QAM	1/2
4	16QAM	3/4
5	64QAM	2/3
6	64QAM	3/4
7	64QAM	5/6
8	256QAM	3/4
9	256QAM	5/6

Figure 1: MCS Values with corresponding Modulations and Coding Rates

We can have a better estimation of the most appropriate MCS by considering:

- Signal to Noise Ratio(SNR)
- Packet Error Rate (PER)

In the original algorithm, MCS value is approximated by comparing the current SNR of the received packet with predefined threshold values. The threshold values are stored in a array variable called "threshold".
For shifting the MCS value up, we compare the current SNR(stored in "snrInd" variable) with snrUp variable which stores the threshold variable values with a new last value, infinity (to compare with all values more than the maximum value of the threshold variable array). The values of the snrUp can be adjusted by using the rcsAttack variable($snrUp = threshold + rcsAttack$), this is done to adjust the sensitivity of increment in the MCS values.
A similar logic is applied for shifting MCS values down. we compare the current SNR(stored in "snrInd" variable) with snrDown variable which stores the threshold variable values with a new first value, -infinity (to compare with all values less than the minimum value of the threshold variable array). The values of the snrDown can be adjusted by using the rcsRelease variable($snrDown = threshold - rcsRelease$), this is done to adjust the sensitivity of decrement in the MCS values.

3. Algorithm

In our algorithm we did not modify or use any part of the previous algorithm.

Our focus was to first check the successful decoding of the packet to ensure the reliability of the link for a specific MCS, and then check the change in SNR, so as to determine the next MCS scheme.

In the beginning of the algorithm we associate MCS values with SNR of current packet that has been received. Corresponding to the SNR value of the first packet, a value is assigned to the MCS. To select appropriate SNR value blocks corresponding to the MCS value we referred to [1]. Using this logic we can directly start with

a more approximate MCS value rather than starting from $MCS = 0$ which has been followed in the original algorithm.

We consider four cases w.r.t. SNR and PER to change MCS(snrInd is later equated to MCS):

- SNR decreases and we get a packet error
  - decrease snrInd by two, as there is a packet error.
- SNR decreases but we dont get a packet error
  - decrease snrInd by two.
- SNR increases but we get a packet error.
  - decrease snrInd by two, as there is a packet error.
- SNR increases and we dont get a packet error.
  - decrease snrInd by one.

```
Initializations
for loop of simulation do
    SNR simulations generated ;
    PER generated;
    if packet number == 1 then
        Compare with the if-else conditions to assign the first MCS value corresponding to SNR, from [1];
    else
        for all other packets keep assigning the previous MCS value;
    end
    delta= difference between current pkt SNR and previous pkt SNR;
    if packet received does not contain any errors then
        if delta >0 then
            increase snrInd by 1;
        end
        if delta < 0 and |delta|>1 then
            decrement snrInd by 2;
        end
    else
        Since packet received has errors, decrement the snrInd by 2;
    end
    if snrInd >= 9 then
        if current channel badwidth = CBW20 and number of users = 1 then
            snrInd=8 (maximum allowed MCS value for this configuration);
        else
            snrInd=9;
        end
        if snrInd<0 then
            snrInd=0 (minimum allowed MCS value is 0);
        end
    end
    MCS=snrInd;
end
```

Algorithm 1: Rate Control Algorithm

4. Results

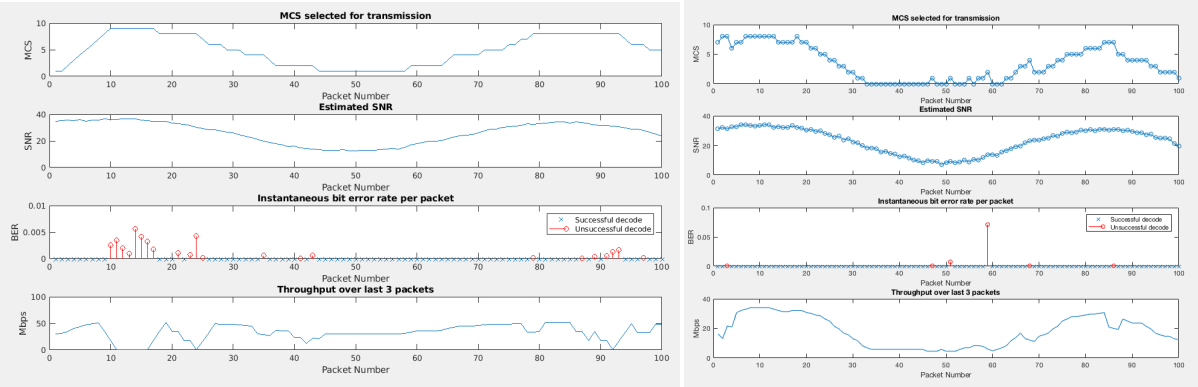
As discussed in the algorithm, we try to balance the BER for every input data by increasing/decreasing the MCS value accordingly and thereby maintaining the average throughput.

Table 1: Readings for Different Scenarios

Cases	Original Algorithm		New Algorithm	
	Throuhput(Mbps)	BER	Throughput(Mbps)	BER
Orignal Conditions	22.57	0.08	20.19	0.04
SNR amplitude =14	16.96	0.26	16.28	0.20
MaxJump(SNR) = 1.5	21.63	0.12	20.35	0.03
SpatialMapping				
Hadamard	22.57	0.08	20.198	0.04
Fourier	22.57	0.08	20.198	0.04
Custom	22.57	0.08	20.198	0.04
Model D(CBW20)	14.28	0.06	12.2	0.04
Model D(CBW80)	35.91	0	35.50	0
Model D(CBW80)	35.91	0	35.50	0
Model E(CBW20)	13.33	0.13	11.9	0.06
Model E(CBW40)	26.58	0.08	23.26	0.02
Model E(CBW80)	36.73	0.02	32.75	0.04
Model F(CBW20)	ERROR	ERROR	12.22	0.02
Model F(CBW40)	27.77	0.04	22.08	0.04
Model F(CBW80)	32.84	0.22	31.19	0.02

We have considered a corner case in which the first case is NaN, i.e., no packet is received. Here our algorithm ignores the current packet.

The results covers all the possible cases including the boundary cases which can not be covered by the original algorithm in CBW20, Model-F configuration:
Error using coder.internal.error (line 14)
The combination of ChannelBandwidth = 'CBW20', NumSpaceTimeStreams = 1 and MCS = 9 for user 1 is not supported.



(a) Original Algorithm(CBW80, Model F)

(b) New Algorithm(CBW80, Model F)

Figure 2: Scenario: Model F, CBW80

5. Conclusion

Results shown in the Table 1 prove that the new algorithm reduces the bit error rate considerably while maintaining the average throughput of the signal.

Bibliography

[1] http://www.enterprisenetworkingplanet.com/netsp/article.php/3747656/WiFi-Define-Minimum-SNR-Values-for-Signal-Coverage.htm