Rate Control Algorithm Comparison
NS3 Project - Wireless Networking

Renzo Arreaza 4567560
Marcelo Guerrero 4736605

April 2018

1 Introduction

Different data rates are available for each 802.11 standard. It is the job of the rate control algorithm to dynamically determine the rate that maximizes the throughput according to the channel conditions. Many types of algorithms can be found on commercial devices and on research documents. Most of them are based on the received signal power and statistics of transmissions.

In this project, we will compare the performance of the IdealWifiManager (based on signal strength), the MinstrelHtWifiManager (based on statistics of transmissions) and the ConstantRateWifiManager, on a 802.11n network. The simulation environment does not have obstacles and makes use of the Log Distance Propagation Loss model (no fading channel).

2 Algorithms

2.1 IdealWifiManager

The IdealWifiManager algoritm uses the SNR of the previous packet to select a transmission mode based on a set of SNR thresholds. The SNR value at the receiver side is send to the transmitter via an ACK packet. The SNR threshold table is build from a target BER and mode-specific SNR/BER curves.

2.2 MinstrelHtWifiManager

MinstrelHT is based on the Minstrel algorithm. It dynamically probes the channel to determine what rate can offer a better throughput. It was developed specially for 802.11n/ac standards. This algorithm arranges the combinations of channel width, spatial streams, guard intervals, modulation and coding rate, in Modulation and Coding Schemes (MCS) groups. For each group, it keeps several statistics such as the maximum and average throughput, and successful and failed transmissions. These statistics are updated per each A-MPDU transmitted and are used to calculate the values of the Multiple Rate Retry (MRR) chain. There are three rates on this chain. These are maxThrRate (maximum throughput rate), secondmaxThrRate (second maximum throughput rate) and maxProbRate (maximum probable rate). The algorithm first uses the maxThrRate and when an A-MPDU needs to be re transmitted (BlockACK timeout) the next value on the chain is used. The MRR chain is updated according to an update interval that can be defined by the user. Differently from legacy Minstrel, MinstrelHT does not use the "look around mechanism", but makes sure to

sample all rates at least one on each interval. However, high data rates have preference over low rates and high probability of error rates [1].

2.3 ConstantRateWifiManager

This algorithm uses a constant rate for data and RTS transmissions. Was included in this test for comparison purposes.

3 Implementation

For this project, we implemented a 802.11n network using the following configuration. This configuration can only be changed by modifying the source code.

- PHY: 802.11n / Freq: 5.15 Ghz
- Propagation Model: Log Distance Path Loss / Exp: 3 / Reference Loss: 46.677 db at 1 m.
- ErrorModel: NistErrorRateModel
- STA Active Probing: False
- Energy Detection Threshold (receiver sensitivity): -96 dbm
- Tx Power: 16 dbm
- Tx Rx Antenna Gain: 0 db
- GreenFieldEnable: False
- HT Support: Enable / 802.11e QoS: Enable (only best effort traffic) / A-MPDU: 65535
- Rx Noise: -93 dbm
- STA Rate control algorithm: MinstrelHtWifiManager

In the application, an AP constantly sends packets to a STA. The AP is elevated 3 meters from the ground and sends data with a rate of 100 Mb/s. During the execution of the program, the STA moves away from the AP up to a certain point and then starts to move back to its original position. The STA moves in steps and in each step, it stays for a certain period of time. It is possible to add 4 additional STAs to the application. For this case, the AP also sends packets to these devices using the same data rate. These devices are located 5 meters away from the AP and are elevated 1 meter from the ground. This position does not change during the execution of the application. These STAs are only used to add load to the AP and interference to the channel.

During the simulation, the transmission rate of the AP used towards the STA is continuously stored in a Gnuplot dataset. This rate is calculated by the control rate algorithm. In each step of the STA, the throughput received by the STA is calculated and stored in a Gnuplot dataset. The amount of failed packets transmitted by the AP towards the STA are also stored in a Gnuplot dataset. These datasets are used to create gnuplot-ready plotting commands files to be used by the Gnuplot application.

The application was created to measure the performance of the MinstrelHT, Ideal and Constant rate algorithms. It is possible to test other rate algorithms by performing small modifications to

the source code. Note that this code can easily be adapted to evaluate other scenarios such as QoS, Mixed technologies, etc.

The following parameters can be used during the execution of the application. If a parameter is not defined during the execution, its default value is used.

- initialDistance: Initial distance of the STA / Default = 1 m
- shortGuardInterval: Enable short guard interval / Default = false
- spatialStreams: Number of spatial streams / Default = 1
- rtsThreshold: RTS Threshold / Default = 65535
- apRateControl: Rate control Algorithm of the AP / Default = MinstrelHtWifiManager
- dataConst: Rate of the date plane for Constant Rate Algorithm / Default = HtMcs7
- channelWidth: Channel width of the stations / Default = 20 Mhz
- steps: How many different steps to try / Default = 45
- stepsTime: Time on each step / Default = 1 s
- stepsSize: Distance between steps / Default = 1 m
- additionalUsers: Add 4 additional users / Default = false

4 Simulation

Since there are many combinations of parameters that could be evaluated, we decided to focus on the set of combinations with the highest and lowest values of the modulation and coding schemes of 802.11n. The following scenarios were evaluated:

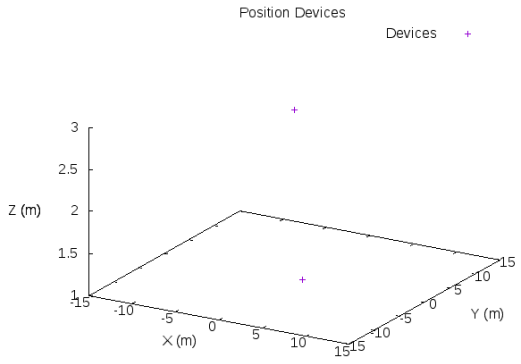
1. Spatial streams: 1 / Channel Width: 20 MHz / GI: 800 ns (long)
2. Spatial streams: 4 / Channel Width: 40 MHz / GI: 400 ns (short)

Both scenarios were evaluated without additional users. The time that the STA stays in each step was configured to 5 sec. The other parameters were set to their default values. Note that in 45 meters, the path loss is equal to -96.27 dbm (this value was calculated using the well-known log distance path loss formula). This value is slightly smaller than the rx sensitivity of the AP. However, the transmission power is 16 dbm, therefore, we expected to receive some packets in this position.

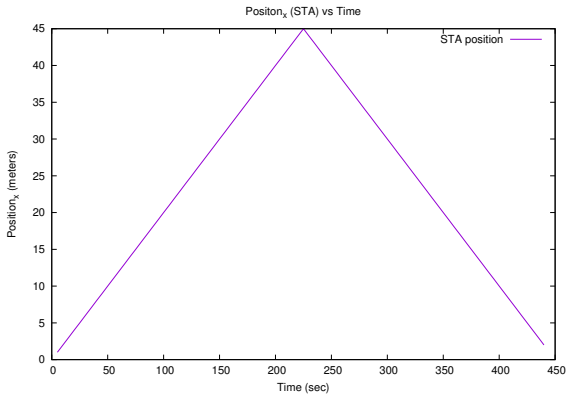
The parameters used by the rate control algorithms during the simulation are listed below.

1. IdealWifiManager. BerThreshold = 1e-05
2. MinstrelWifiManager. UpdateStatistics = 100 ms
3. ConstantRateManager. DataMode = HtMcs3 (26 Mb/s), HtMcs4 (39 Mb/s) and HtMcs7 (65 Mb/s)

Figure 1 presents the initial position of the devices (AP and STA) and the trajectory of the STA.



(a) Initial position of AP and STA



(b) Position over time of STA

Figure 1: Device positions

5 Performance Analysis

Before starting with the analysis of the results, it is important to recall that the transmission power of the AP remains fixed during the simulation. Therefore, if two signals, modulated with different schemes, are transmitted from the AP, the signal with the lowest modulation will be more robust against interference and noise. Additional to this, from the link budget previously calculated, we found that the signal strength at 45 m is -80.27 dbm. Hence, we expect to be able to receive packets at this distance since the rx sensitivity is -96 dbm. However, NS 3 also keeps track of the interference on the channel. This is calculated based on the modulation and coding scheme of the packets, error model and strength of the signal. As a consequence, packets can be destroyed during collisions or its SNIR can be drastically reduced before 45 m.

5.1 Scenario 1

Figures 2, 3 and 4 show the throughput, rate and failed packets obtained in this scenario. First, by looking at figure 2a, we are able to see that the Ideal algorithm manages to maintain a connection throughout the entire test, with a minimum throughput value of 16.681 Mb/s. The throughput started to get affected at around 100 seconds which from figure 1, corresponds to 20 meters which in turn corresponds to a path loss of 85.71 dbm. The lowest region of throughput started at around 180 seconds which corresponds to 36 meters and path loss of 93.37 dbm. It is interesting to see that the rate decreased and increased symmetrically. It went from 65 Mb/s to 26 Mb/s, and then back to 65 Mb/s. As we mentioned previously, the Ideal algorithm does not let the flow go below a certain BER (in this case 1e-05), this can be seen in figure 4a. When the amount of failed packets started to increase, the rate was changed to decrease this behavior. Note that none of the peaks were over 3500 packets and the duration of the peaks is very short (small width). When the strength of the signal increased, the rate was incremented and some packet loss was seen on the connection. However, it is always good to have a small amount of packet loss since this means that the channel is not being under utilized. This packet loss was reduced when the STA moved closer to the AP.

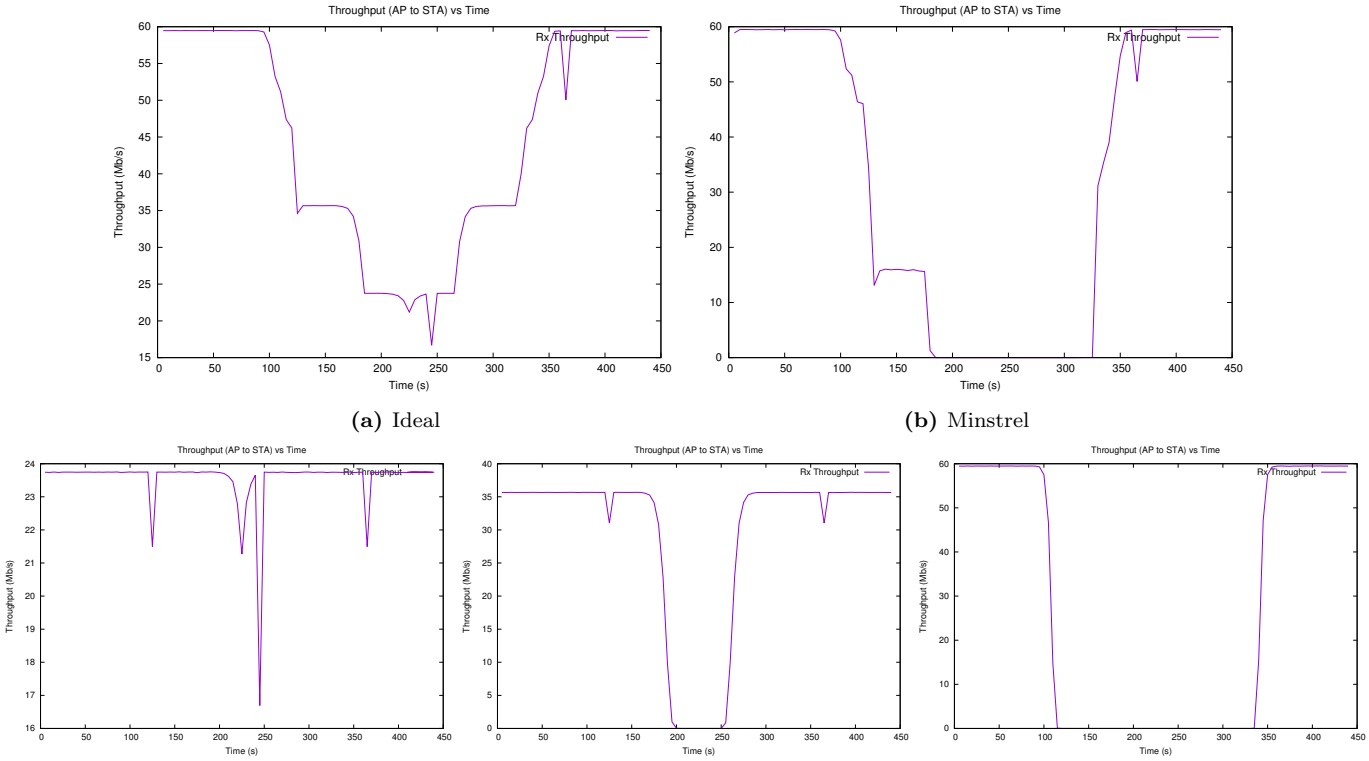
Contrary to the Ideal algorithm, the MinstrelHt algorithm completely lost connection for a period of about 145 seconds. Specifically, from around 180 seconds to around 325 seconds which corre-

sponds to 36 meters and 25 meters respectively. This can be observed in 2b. The throughput of this algorithm also started to decrease around 100 seconds. This is expected since both algorithms start with 65 Mb/s which is achieved by using 64 QAM, 5/6 coding rate and 800 GI (see the initial discussion of this section). One key aspect of the results obtained with this algorithm is that none of the graphs are symmetric. It takes a while to recover after the user is within a range that previously worked. The changes on rate of this algorithm is presented in figure 3b. From our previous discussion about the operation of this algorithm, we know that the MRR chain uses three values. These are maxThrRate, secondmaxThrRate and maxProbRate. We also know that all supported rates can be sampled and that high rates have preference over low rates. Initially, maxThrRate was set to 65 Mb/s. This rate was maintained during the first 100 seconds (20 meters). Since there was no packet loss in this period, all transmissions were sent with this rate and every transmitted frame improved the statistics of this group (Mcs7). None of the remaining rates were sampled within this period. In the following 25 seconds (5 meters), the packet loss slowly started to increase. When the first A-MPDU frame needed to be re transmitted, the secondmaxThrRate, which is set to 58 Mb/s, was used. The algorithm transmitted frames with this rate during approximately 9 seconds regardless that some frames were lost, meaning that maxThrRate, secondmaxThrRate and maxProbRate were set to 58 Mb/s. After this time, at 110 seconds, the algorithm set again maxThrRate to 65 Mb/s. Due to packet loss, the A-MPDU frames were transmitted using 65 Mb/s and 58 Mb/s and eventually, the maxProbRate, which was set to 52 Mb/s, had to be used. Once again, the algorithm sent frames using this rate for about 15 seconds.

At 125 seconds (25 meters), the amount of failed packets rapidly increased reaching a value of roughly 18000 frames. This amount of frame loss remained high until 180 seconds (36 meters) when the throughput went to 0. This can be seen in 4b. This poor performance was due to the nature of this algorithm of giving priority to high rates over low rates. It is possible to observe in figure 3b that the rate was very unstable during this period. maxThrRate was always set to 65 Mb/s and secondmaxThrRate was switching between 52 Mb/s, 39 Mb/s, 26 Mb/s. In some occasions, the algorithm sampled low rates such as 13 Mb/s and 6 Mb/s, however, these rates were never considered as possible values for secondmaxThrRate or maxProbRate. The Minstrel algorithm was implemented with the idea that in bad channel conditions, frames transmitted with high rates have more probability of being successful (with bursty noise, the change of a collision is smaller). During the time that the throughput was 0, the only considered rates were 65 Mb/s and 58 Mb/s. This confirms our previous statement. This algorithm behaves differently when the channel goes from a bad condition to a good condition. The throughput was expected to increase around 270 seconds (36 meters), however, since only high rates were being considered, it was not possible to receive a Block ACK frame. Finally, at 300 seconds (30 meters), the algorithm decided to sample a low rate, specifically 6 Mb/s. Since the frame was successfully transmitted and because high rates groups had very poor statistics, the algorithm decided to set 6 Mb/s as secondmaxThrRate. The AP transmitted frames using 65 Mb/s and 6 Mb/s during about 25 seconds. At 325 seconds (25 meters), the algorithm sampled a rate between these two values (39 Mb/s) and since the frame was successfully transmitted, the three values of the MRR chain were set to 39 Mb/s. The algorithm increased its rate following the same pattern until reaching again 65 Mb/s.

Finally, figures 2c, 2d, 2e, 4c, 4d and 4e, present the throughput and failed packets of the ConstantRate algorithm for the MCS groups 3 (26 Mb/s), 4 (39 Mb/s) and 7 (65 Mb/s). These algorithms can be used to see how the throughput would have behaved if the Ideal algorithm would not have changed the rate at certain points. These results confirm the good performance of the Ideal algorithm. It is nice to notice that despite the completely different approaches of the algorithms, both have very similar results of throughput, in fact, the results are identical on periods that overlap, such as the first 100 seconds of the Ideal and the Constant Rate with Mcs7. The

behavior of the channel depends on the transmission power, modulation scheme, coding rate, error model and interference level. If we do not modify any of these parameters (as in this case), the results will be identical (see the discussion presented at the beginning of this section). The local minimums in the $Mcs = 3$ and $Mcs = 4$ plots at the 125 and 365 seconds mark are due to internal behaviour of the code and cannot be explained. It is important to mention at this moment the limitations of simulation tools compared to real environments. In a real environment, there will be more randomness affecting the channel including of course multipath fading.



(a) Ideal

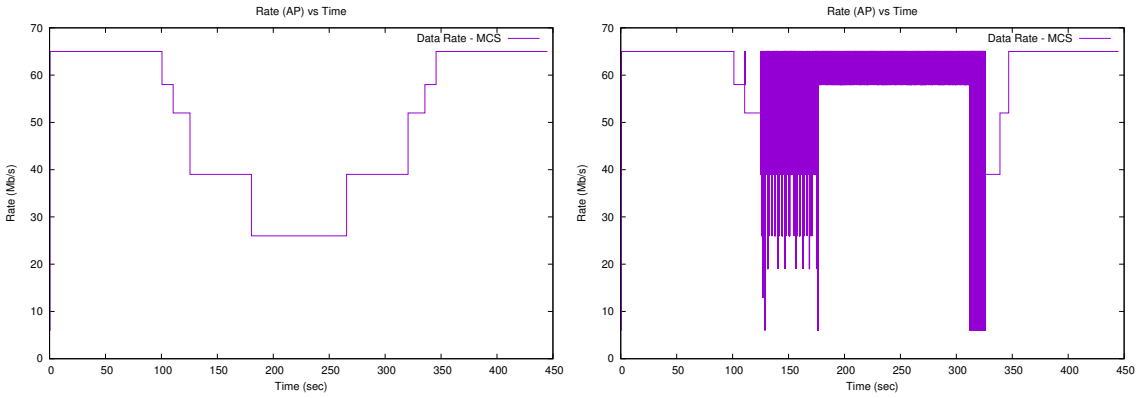
(b) Minstrel

(c) Mcs 3

(d) Mcs 4

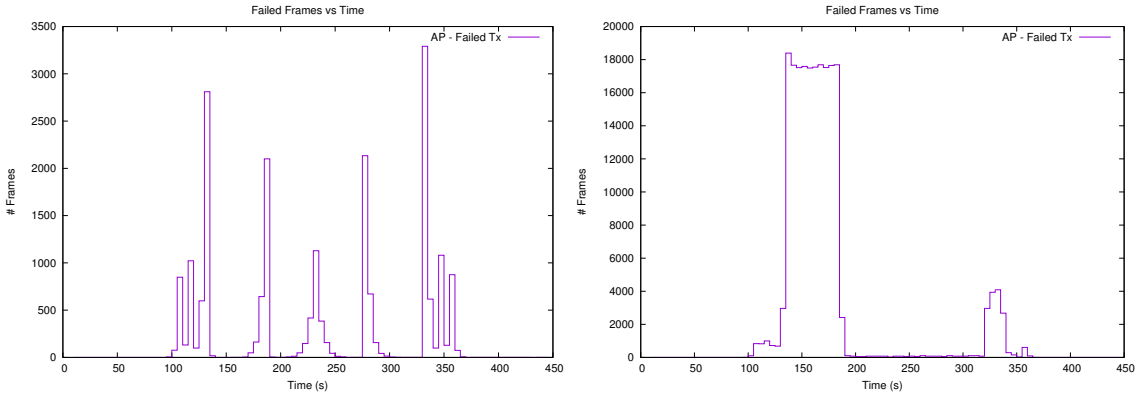
(e) Mcs 7

Figure 2: Throughput vs Time- 20Mhz, Long guard interval, 1 spatial stream, 0 additional users

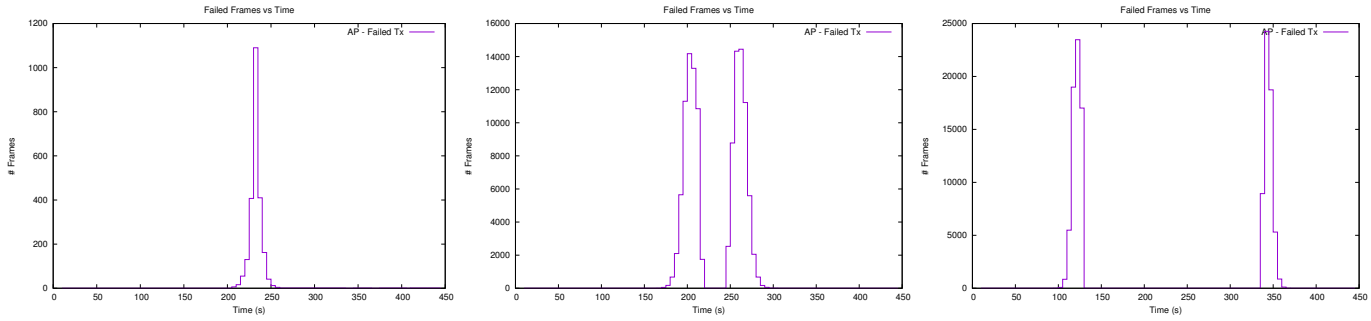


(a) Ideal **(b)** Minstrel

Figure 3: Rate vs Time - 20Mhz, Long guard interval, 1 spatial stream, 0 additional users



(a) Ideal **(b)** Minstrel



(c) Mcs 3 **(d)** Mcs 4 **(e)** Mcs 7

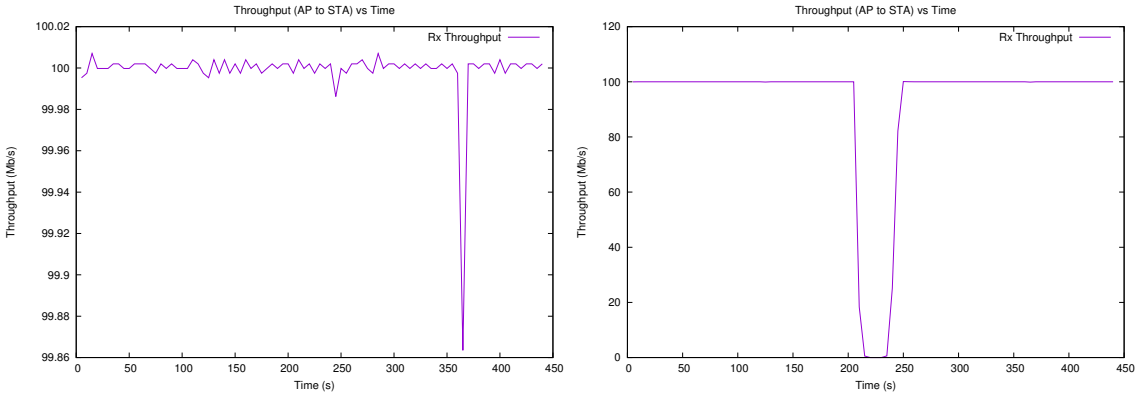
Figure 4: Failed Packets vs Time - 20Mhz, Long guard interval, 1 spatial stream, 0 additional users

5.2 Scenario 2

In this scenario, we only compared the performance of the Ideal and MinstrelHt algorithms. Figures 5, 6 and 7, present the results of throughput, rate and failed packets respectively. The throughput obtained with the Ideal algorithm, figure 5a, was around 100 Mb/s during the whole simulation.

There was a low peak at 18 meters that reached a value a bit above of 99.86 Mb/s. This behavior, as well as, the packet loss (only 1 packet at far and close distances from the AP) observed in figure 7a, cannot be explained, and hence it is assumed that it is due to internal procedures of NS3. During the whole simulation, the rate was above 100 Mb/s. The rate in this case also changed gradually to not overpass the imposed BER.

The MinstrelHT algorithm also had a disconnection on this scenario. However, the amount of time that throughput went to 0 was smaller compared to that in the previous scenario. It was around 225 seconds at a distance close to 45 meters. This can be observed in figure 5b. Once again, only very high rates were considered on this scenario, namely 600 Mb/s and 540 Mb/s (figure 6b). For this reason, at 200 seconds (40 meters) the amount of failed transmissions, presented in figure 7b started to increase and the throughput rapidly decreased to 0 (a detailed explanation as to why this algorithm has this behavior can be found in the discussion of scenario 1). One can be tempted to believe that the MinstrelHt algorithm performs better when very high data rates are used. However, the better performance on this scenario is due to the channel is being sub-utilized. As we mentioned previously, the AP sends packets using a data rate of 100 Mb/s and thanks to the spatial streams, guard intervals and channel width of this scenario, the channel can easily handle rates from 60 Mb/s up to 600 Mb/s. If we increase the data rate of the AP to a value equal or greater than 600 Mb/s, the performance on this scenario would be very similar to the performance obtained on the scenario 1 (the reader can easily prove this statement).

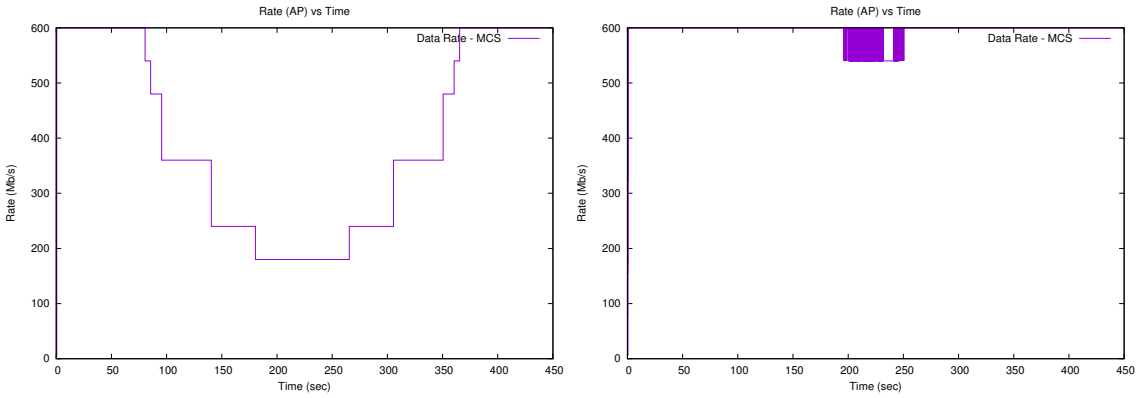


(a) Ideal **(b)** Minstrel

Figure 5: Throughput vs Time - 40Mhz, short guard interval, 4 spatial stream, 0 additional users

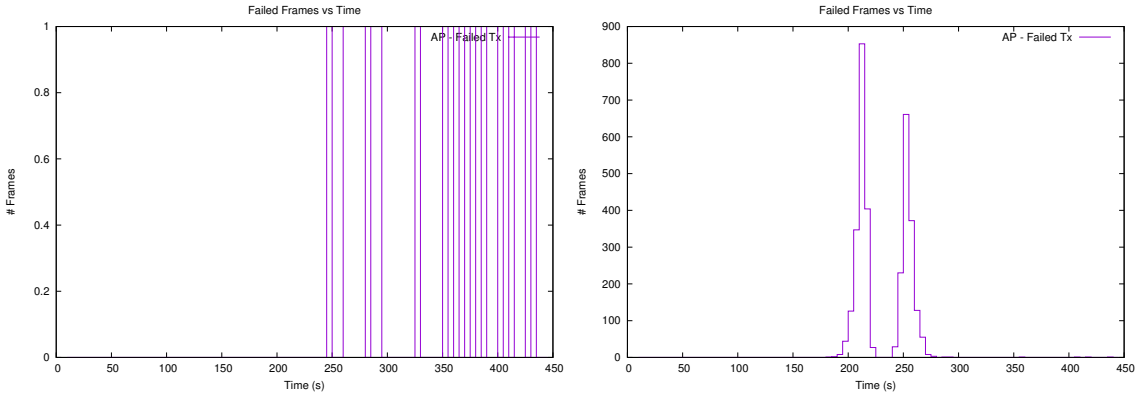
5.3 Extra Scenarios

As was described in section 3, it is also possible to enable 4 extra users. The results of which can be found in [2]. We decided not to include them in this report because it still needs some work. Most of the measurements are done from the AP, and are therefore a mix of all the stations. For this reason it isn't possible to interpret the data correctly. Further research could be done by improving this part of the simulation and seeing what influence the extra users have on the performance of the algorithms.



(a) Ideal **(b)** Minstrel

Figure 6: Rate vs Time - 40Mhz, short guard interval, 4 spatial stream, 0 additional users



(a) Ideal **(b)** Minstrel

Figure 7: Failed Packets vs Time - 40Mhz, short guard interval, 4 spatial stream, 0 additional users

6 Conclusions and future work

In an environment with no obstacles and no multipath fading, the performance of the Ideal algorithm clearly exceeded the performance obtained with the MinstrelHt and ConstantRate algorithms. The changes on the transmission rate determined by this algorithm, are reflected on the good overall performance of the connection. The algorithm indirectly tries to avoid disconnections by making use of very low rates when the conditions of the channel are deteriorated. Also, the BER imposed by this algorithm leads to having a small amount of failed transmissions with peaks of short duration. The assumptions made by the ideal algorithm (channel deterioration because of distance, and decreasing SNR) are true in this test.

The performance of the MinstrelHt algorithm is very poor in an environment with no obstacles and no multipath fading. The tendency of using high transmission rates when the channel is not optimal, leads to obtaining high frame loss for large periods of time which greatly deteriorates the throughput of the connection. And even worse, it leads to very large periods of total disconnection. Here, the assumptions are not met. The decrease in channel quality is because of distance and

decreasing SNR and not because of more interference. Therefore, the approach the algorithm takes is not a good solution.

The way that the MinstrelHt algorithm operates, makes us believe that its performance can be improved in an environment with obstacles and multipath fading. A user moving in this type of environment may be located in places with bad channel conditions for short periods of time. Therefore, not switching to low rates during these periods might be beneficial to the flow once the channel improves. It can also subjectively force users to stay close to the AP and avoid places with bad channel conditions since the performance is dramatically deteriorated in these places. It is important to mention that these statements are simply assumptions and it is necessary to corroborate them with a new simulation. Starting from this application, the building class can be used to create an environment with obstacles and the Spectrum class can be used to simulate a PHY with multipath fading.

References

[1] ns-3: ns3::minstrelhtwifimanager class reference. `https://www.nsnam.org/docs/release/3.27/doxygen/classns3\_1\_1\_minstrel\_ht\_wifi\_manager.html`.

[2] Renzo Arreaza and Marcelo Guerrero. Tss1-ns3 repository. `https://github.com/renzoarreaza/TSS1-NS3`.