University of Southern California

EE597 Project – Saturation Throughput

Wireless Networks

by

Wanwiset Peerapatanapokin

Jiang Zhang

Table of Contents

Project Overview

This project provides analysis and simulation of 802.11 Wi-Fi DCF saturation throughput based on Matlab and NS-3. Specifically, the analysis of the 802.11 Wi-Fi DCF saturation throughput heavily follows the paper written by G. Bianchi ("IEEE 802.11-saturation throughput analysis," in *IEEE Communications Letters*, vol. 2, no. 12, pp. 318-320, Dec. 1998). We will start off by introducing and defining the necessary technical terms, then move on to mathematical analysis. Finally, we will present a graph that relates saturation throughput to the number of contending stations, along with varying some WIFI DCF parameters.

Part 1 – Analysis of 802.11 WIFI DCF Saturation Throughput

1.1 Introduction of 802.11 WIFI DCF

IEEE has introduced 802.11 as standard protocol for WIFI. This standard uses CSMA/CA (carrier sense multiple access with collision avoidance) technique as its MAC or medium access control protocol. Specifically, it uses a CSMA/CA method with binary exponential backoff called Distributed Coordination Function or DCF. In this analysis, we focus on the saturation throughput which is defined as the limiting throughput of a channel as the load increases in stable conditions. We observe the throughput as the number of contending stations increases. Also, we assume that each station always has a transmission waiting in queue, hence, saturation.

In the DCF method, a station waiting to transmit will monitor the channel activity. If the channel is idle for a period equal to a distributed interframe space (DIFS), the time following an idle DIFS is slotted. Each station is only allowed to transmit at the beginning of a slot. Once an idle DIFS is detected a random backoff time is generated for each station waiting to transmit. For every idle slot, the backoff time is decremented. When the backoff time of a station reaches zero, it transmits its packet. Note that when the channel is active, the backoff timer doesn't decrease. Fig. 1. Represents the visualization of DCF mechanism.

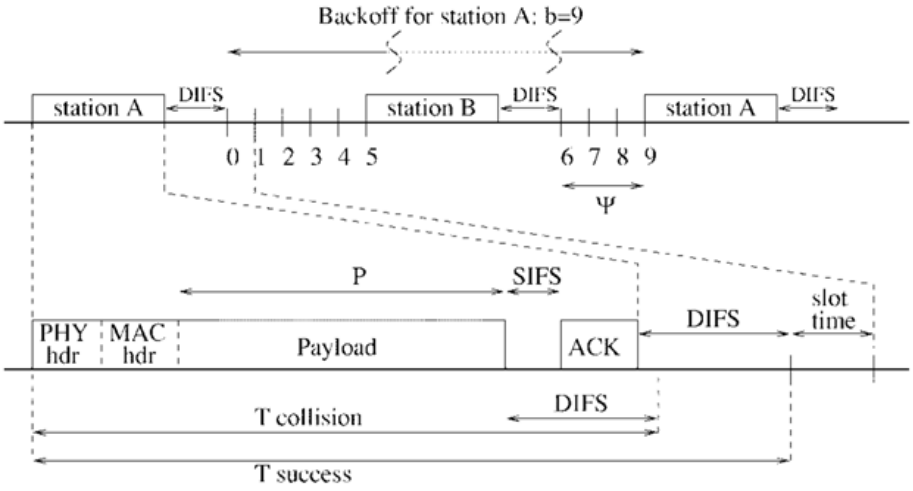

Fig. 1. Basic access mechanism.

After a successful transmission (ACK received) a backoff timer is generated in a uniformly random range between (0, *W*-1), which the station waits before it transmits the next packet. If the transmission fails due to collision with another station and ACK not received, the maximum window is doubled to (0, 2*W*-1), until the maximum value of *W**2^*m*. Here, *W* is defined as the minimum backoff window and *m* is the maximum backoff stage. Fig.2. Below is a discrete-time Markov chain representation of DCF states of a transmission station with *p* representing the collision probability.

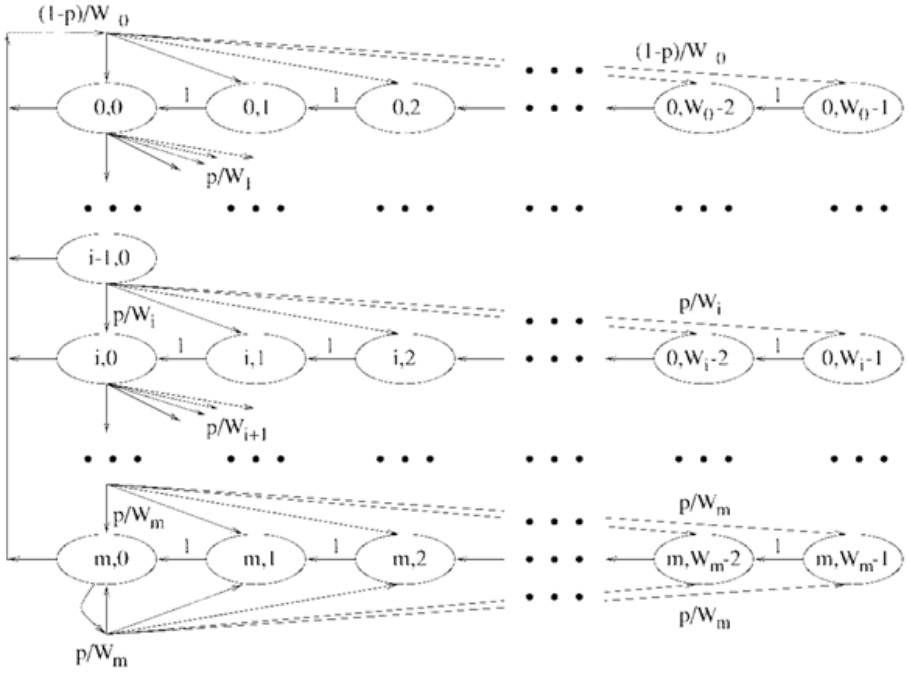

Fig. 2. Markov chain model

1.2 Saturation Throughput of 802.11 WIFI DCF

Again, the mathematical analysis follows Bianchi's IEEE 802.11-saturation throughput analysis. From the Markov chain and the transition probabilities, we can obtain the following steady-state equations.

$$b_{i,0} = p^i b_{0,0}, \qquad i \in (0, m-1)$$

$$b_{m,0} = \frac{p^m}{1-p} b_{0,0}$$

$$b_{i,k} = \frac{W_i - k}{W_i} b_{i,0}, \qquad k \in (0, W_i - 1).$$

$$1 = \sum_{i=0}^{m} \sum_{k=0}^{W_i - 1} b_{i,k}$$

Solving the equations, we obtain:

$$b_{0,0} = \frac{2(1-2p)(1-p)}{(1-2p)(W+1)+pW(1-(2p)^m)}. \qquad (1)$$

Then we can find the probability of transmission τ, which is stages where backoff window is 0

$$\tau = \sum_{i=0}^{m} b_{i,0} = \frac{b_{0,0}}{1-p} = \frac{2(1-2p)}{(1-2p)(W+1)+pW(1-(2p)^m)}. \qquad (2)$$

The final equation is the probability that a packet collides or *p*. Here, (1-τ)^(*n*-1) is the chance that the remaining stations will not transmit and *n* is the total number of transmitting stations.

$$p = 1-(1-\tau)^{n-1} \qquad (3)$$

From (2) and (3) there are two equations and two variables, therefore, it can be solved numerically given the parameters *n, m,* and *W*.

Let's define P_tr as probability that there is at least one transmission in a slot time, then, P_s as the probability that a transmission is successful by the following equations:

$$P_{\text{tr}} = 1-(1-\tau)^n \qquad (4)$$

$$P_s = \frac{n\tau(1-\tau)^{n-1}}{P_{\text{tr}}} = \frac{n\tau(1-\tau)^{n-1}}{1-(1-\tau)^n}. \qquad (5)$$

The number of time slots between two successive transmissions can be defined by:

$$E[\Psi] = \frac{1}{P_{\text{tr}}} - 1.$$

Going back to Fig.1, *T_s*, time used for successful transmission and *T_c*, time used for collision, is defined as follows. Where *H* is PHY header + MAC header and δ is the propagation delay.

$$\begin{cases} T_s^{\text{bas}} = & H + E[P] + \text{SIFS} + \delta + \text{ACK} + \text{DIFS} + \delta \\ T_c^{\text{bas}} = & H + E[P^*] + \text{DIFS} + \delta \end{cases}$$

Finally, the throughput (normalized to 1) is simply the proportion of the time in a successful transmission to the total time of a renewal interval. Here, *E[P]* is the average packet length. In this project it is fixed to 8184 bits or 1023 bytes.

$$S = \frac{E[\text{timeused for successful transm. in interval}]}{E[\text{length of a renewal interval}]} = \frac{P_s E[P]}{E[\Psi] + P_s T_s + (1 - P_s) T_c} \quad (7)$$

Note that the unit of $E[\Psi]$ is in number of timeslots. Hence, $E[P]$, T_c, T_s should be converted to slot times to achieve correct results.

1.3 Numerical Results Simulated by MATLAB

We further continue the analysis by plotting the normalized saturation throughput as the number of transmitting stations increases. To compare with Bianchi's results let us observe 3 cases in particular, W=32 m=3, W=32 m=5, and W=128 m=3. Tab. 1 shows the standard parameters used for 802.11 WIFI.

packet payload	8184 bits
MAC header	272 bits
PHY header	128 bits
ACK length	112 bits + PHY header
RTS length	160 bits + PHY header
CTS length	112 bits + PHY header
Channel Bit Rate	1 Mbit/s
Propagation Delay	1 μs
SIFS	28 μs
Slot Time	50 μs
DIFS	128 μs

Tab. 1. Parameter settings.

Thanks to the MATLAB *fsolve* function, we are able to quickly solve (2) and (3) and plot the normalized throughput. The results, which agree with results in [1], are as shown below.

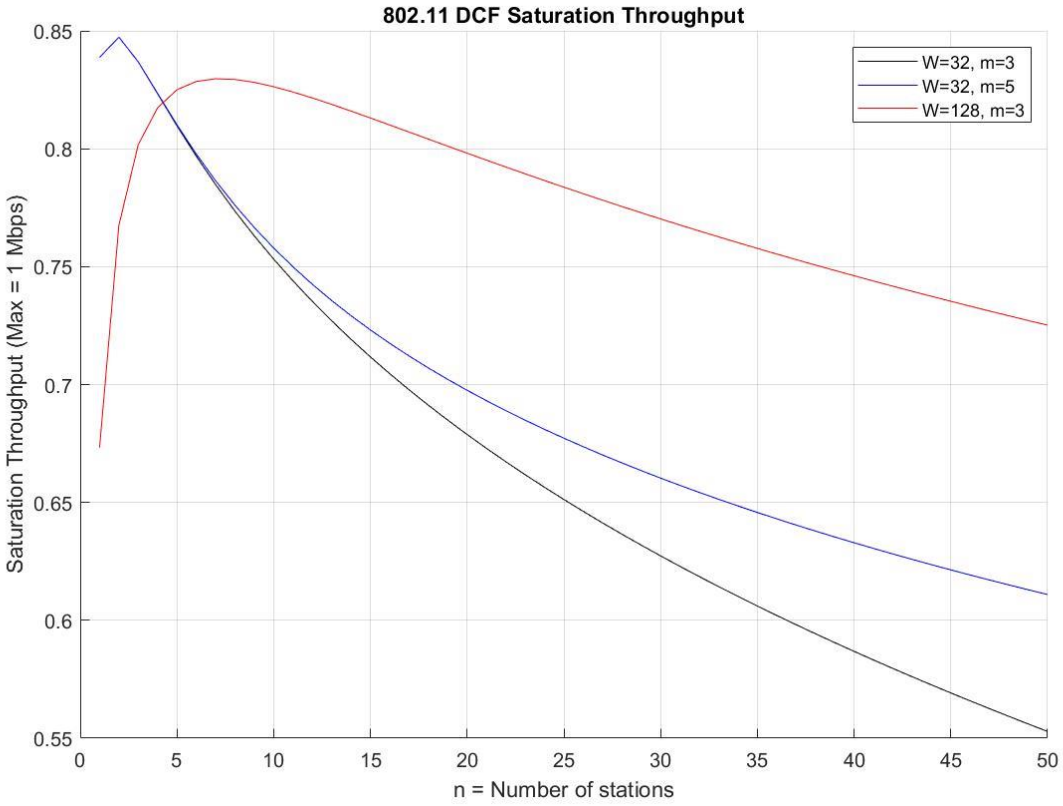

Fig. 3. Matlab simulation results.

Part 2 – Simulation of 802.11 WIFI DCF Saturation Throughput using NS-3

2.1 Introduction of NS-3

The NS-3 simulator is an open, extensible, discrete-event network simulation platform designed primarily for networking research and education. It provides models of how packet data networks work and perform and a simulation engine for users to conduct simulation experiments. In this project, we mainly use the Wi-Fi module of NS-3 for simulation, which is shown in Fig. 4. And we will build the simulation engine as follows:

I. Create Wi-Fi network nodes using `NodeContainer` class;
II. Configure Wi-Fi channel using `YanWifiChannel` class, including propagation loss and delay model;
III. Configure Wi-Fi physical and mac layers using `YanWifiPhyHelper` and `WifiMacHelper` respectively;
IV. Set up Wi-Fi by `WifiHelper` class and install network devices by `NetDeviceContainer` class;
V. Install TCP/IP stack and assign ipv4 address via `InternetStackHelper` class and `Ipv4AddressHelper` class;
VI. Build application layers through `ApplicationContainer` class;
VII. Monitor the network flows and calculate throughput using `FlowMonitorHelper` class.

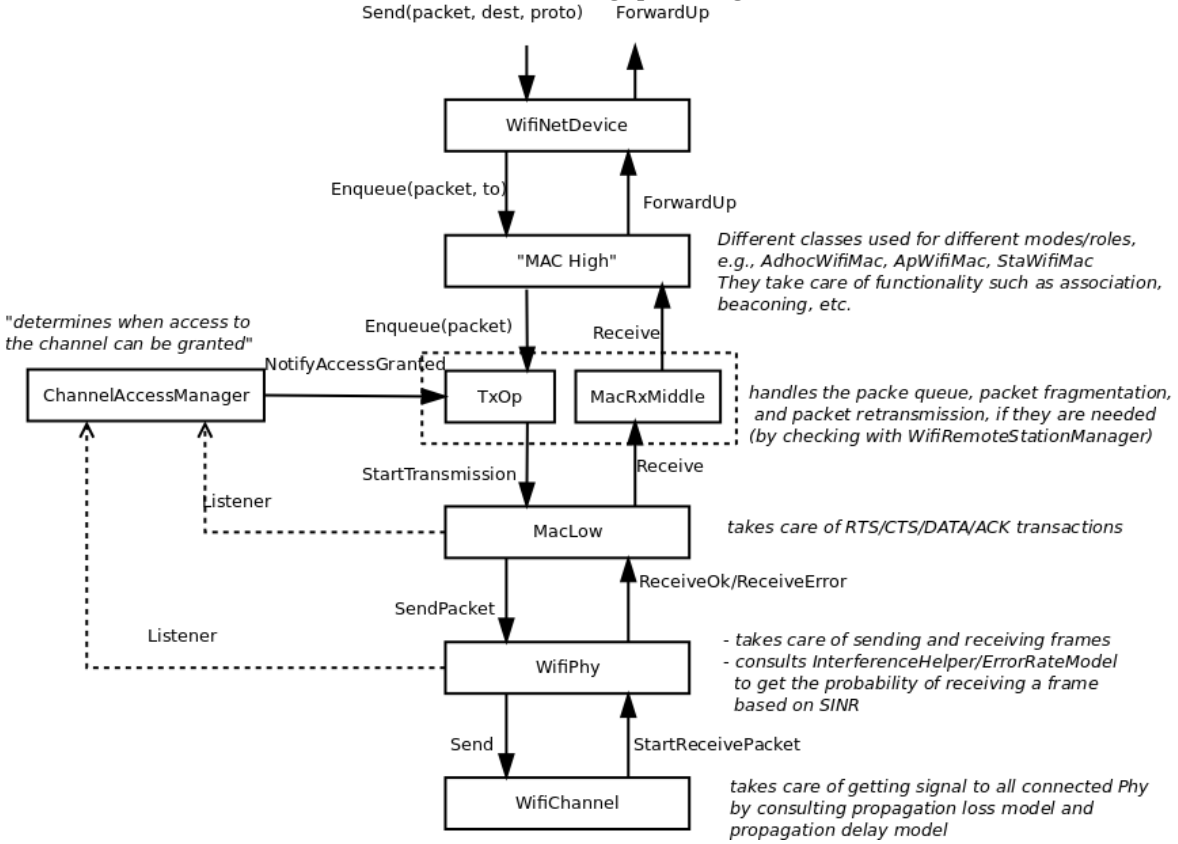

Fig. 4. Wi-Fi modules in NS-3

2.2 Experiment Setup

The topology we implement to simulate the IEEE 802.11 Wi-Fi DCF saturation throughput is shown in Fig. 5 (a), consisting of one access point and N stations.

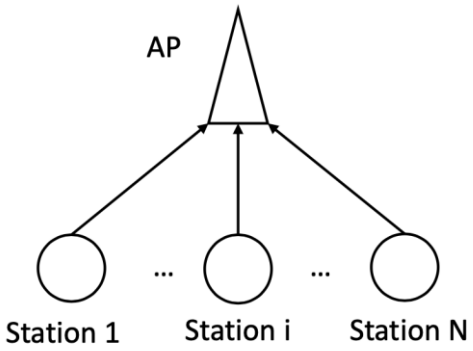

Fig. 5. Network topology.

2.3 Simulation Analysis

Various parameters can be adjusted in the simulation, such as: the number of transmitting stations, the minimum window size (*minCW*), the maximum window size (*maxCW*), the payload size and the simulation time. Due to the simulation being resource and time consuming, we choose to observe values that conform to our previous analysis and Bianchi's paper.

Similar to the analysis, we take 3 cases of minimum and maximum window sizes. W=32 m=3 corresponds to *minCW*=31 *maxCW*=255, W=32 m=5 corresponds to *minCW*=32 *maxCW*=1023, and W=128 m=3 corresponds to *minCW*=127 *maxCW*=1023. For each case, we vary the number of transmitting stations from 1, 5, 10, 15, 20, 30 and 50. We select the simulation time to 100 (seconds) to eliminate randomness in the results as much as possible.

Finally, the payload size is set to 995 bytes, this is due to the NS-3 payload size specifying the data payload inside the UDP application. In Bianchi's experiment, 8184 bits (1023 bytes) is specified as the payload of the MAC layer. Considering that the UDP header is 8 bytes and the IP header is 20 bytes, we get 1023 - (20+8) = 995 bytes as our payload size. Afterwards, in the throughput calculation, we compensate for this and consider 1023 bytes as the payload size. This behavior can be observed in the packet capture of the simulation (as shown in Fig. 6).

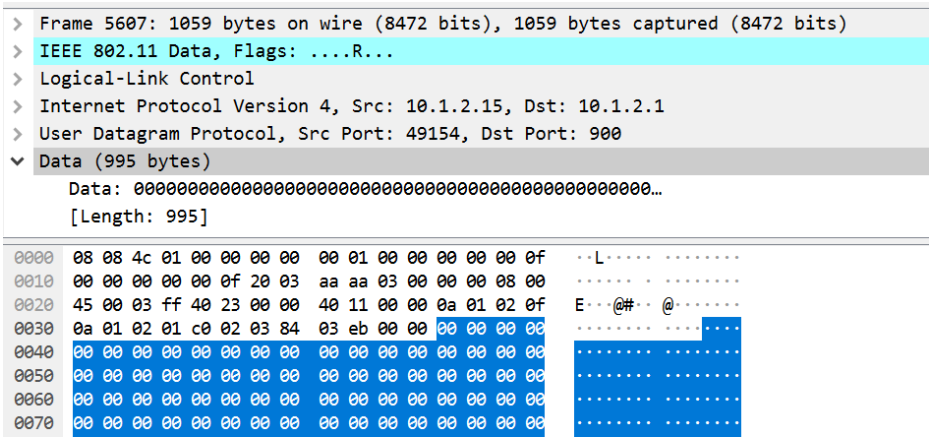

Fig. 6. Packet analysis.

Once the simulation is finished, the output is stored in a text file like below.

nWifi=1
payloadSize=995
minCw=31
maxCw=255
simulationTime=100
Packet count = 10079
Throughput_UDP 0.812288 Mbit/s
Throughput_MAC 0.834865 Mbit/s
****

.
(output omitted)
.

****
nWifi=50
payloadSize=995
minCw=127
maxCw=1023
simulationTime=100
Packet count = 8624
Throughput_UDP 0.69647 Mbit/s
Throughput_MAC 0.715788 Mbit/s
****

We then take the numerical values of Throughput_MAC and plot them in MATLAB alongside the analysis in part 1 of this project. The simulation result, represented with '*,+,x' markers are shown in Fig. 7 below.

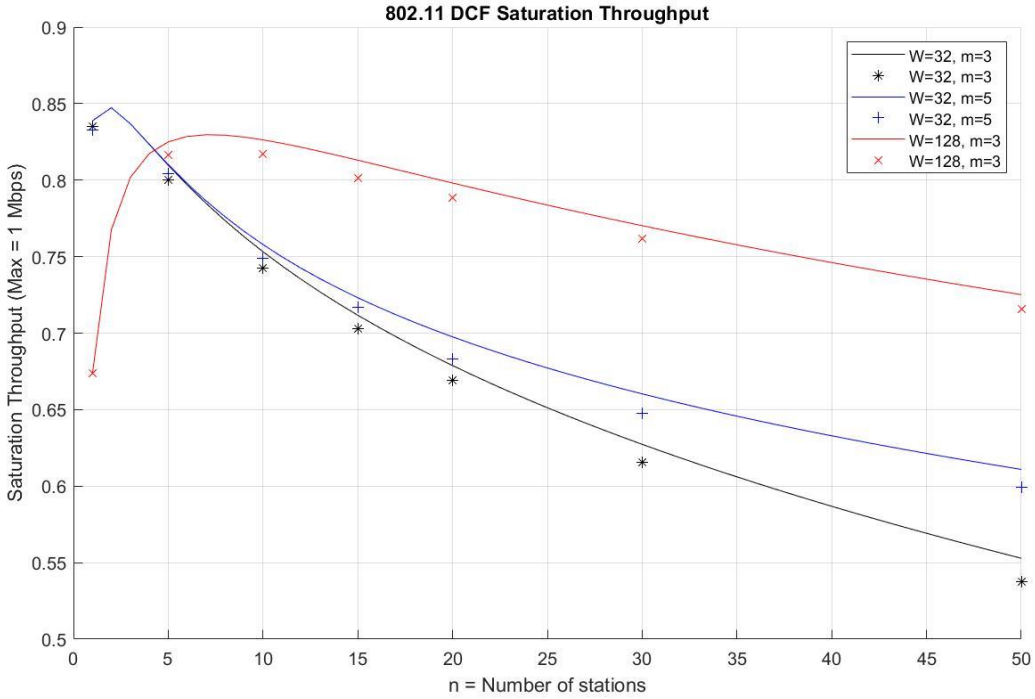

Fig. 7. NS-3 simulation results..

The final plot is very similar to what Bianchi obtained in his paper. The simulation points strongly follows the analysis trend of the DCF saturation throughput as n increases. The average error between the points and the line is 1.28%

2.4 Code Analysis

We revise the source code of NS-3 in order to make the simulated Wi-Fi module configuration more consistent with that talked in [1]. The revised parts are demonstrated as follows:
`wifi-phy.cc` manages the reception of packets and handles collision at the physical layer. Its original behavior will not immediately discard a packet even when there are multiple packets overlapping at the same time. For example, if the Rx senses 3 packets, the 2nd and 3rd won't be received since the Rx is locked on to the first packet. The first packet, however, will be discarded by a random chance with respect to its calculated probability of error from its SINR over time. Fig. 8 demonstrates this behavior.

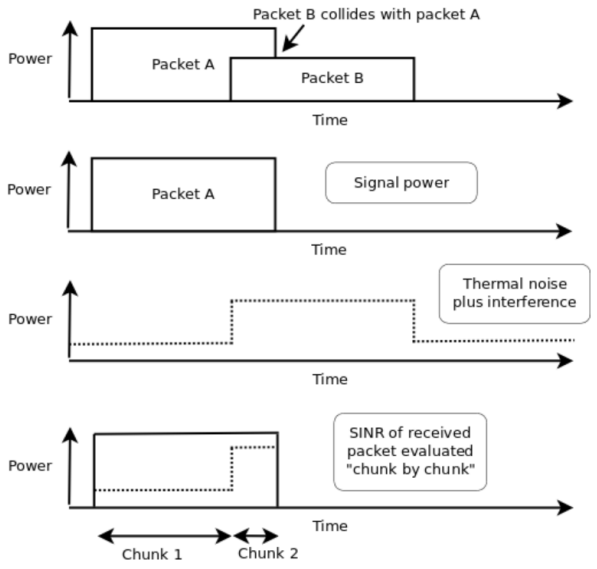

Fig. 8. Collision analysis.

In this project, we are focusing on the MAC layer and assume that any collision will discard all packets involved. So we look to modify this behavior.

// line 34 in `wifi-phy.cc`: create a variable to track collision

```
int collision = 0;  //MODIFIED CODE
```

// line 2453 in `wifi-phy.cc`: increment collision if new Rx arrives while already in Rx state. Also, set the success flag to false.

```
m_plcpSuccess = false;    //MODIFIED CODE
collision++;              //MODIFIED CODE
NS_LOG_INFO("DCF PROJECT LOG: collision=" << collision);
//MODIFIED CODE
```

// line 2521 in `wifi-phy.cc`: only follow through success state if no collision

```
 /*if (m_random->GetValue () > snrPer.per) //plcp reception
succeeded //MODIFIED CODE
    {
      if (IsModeSupported (txMode) || IsMcsSupported (txMode))
        {
          NS_LOG_DEBUG ("receiving plcp payload"); //endReceive
is already scheduled
          m_plcpSuccess = true;
        }
      else //mode is not allowed
        {
          NS_LOG_DEBUG ("drop packet because it was sent using
an unsupported mode (" << txMode << ")");
          NotifyRxDrop (packet);
          m_plcpSuccess = false;
```

```
        }
    }
  else //plcp reception failed
    {
      NS_LOG_DEBUG ("drop packet because plcp preamble/header
reception failed");
      NotifyRxDrop (packet);
      m_plcpSuccess = false;
    }*/  //MODIFIED CODE
      if (IsModeSupported (txMode) || IsMcsSupported (txMode))
        {
         if (collision == 0){
         NS_LOG_DEBUG ("receiving plcp payload"); //endReceive
is already scheduled
          m_plcpSuccess = true;
                }
        }
      else //mode is not allowed
        {
          NS_LOG_DEBUG ("drop packet because it was sent using
an unsupported mode (" << txMode << ")");
          NotifyRxDrop (packet);
          m_plcpSuccess = false;
        }
```

// line 33 in `wifi-mac.cc`: change propagation delay into 1 *us*

```
Time
WifiMac::GetDefaultMaxPropagationDelay (void) {
     //1000m
     // return Seconds (1000.0 / 300000000.0);
     return MicroSeconds (1);
}
```

// line 310 in `wifi-mac.cc`: change SIFS, Slot time, etc.

```
void
WifiMac::Configure80211b (void) {
     NS_LOG_FUNCTION (this);
     SetSifs (MicroSeconds (28));
     SetSlot (MicroSeconds (50));
     SetEifsNoDifs (MicroSeconds (28 + 240));
     SetPifs (MicroSeconds (78));
     SetCtsTimeout (MicroSeconds (10 + 304 + 20 +
GetDefaultMaxPropagationDelay ().GetMicroSeconds () * 2));
     SetAckTimeout (MicroSeconds (10 + 304 + 20 +
GetDefaultMaxPropagationDelay ().GetMicroSeconds () * 2));
}
```

References

1. G. Bianchi ("IEEE 802.11-saturation throughput analysis," in *IEEE Communications Letters*, vol. 2, no. 12, pp. 318-320, Dec. 1998)
2. NS-3 Wi-Fi module documentation: https://www.nsnam.org/docs/models/html/wifi-design.html#interferencehelper
3. Hidden-terminal.cc - general setup reference, create nodes, setup phy, mac, wifi, mobility install IP, routing, onoffhelper, flowmonitor: https://www.nsnam.org/doxygen/wifi-hidden-terminal_8cc_source.html?fbclid=IwAR1tKXmtt2kpp0UZ4ZFpbCIX2wZVGosZC62B8mq4T70cNqLK2gFJp0N1KAA
4. Setting DCF min, max window: https://groups.google.com/forum/#!topic/ns-3-users/TZ6ZftgIkvU
5. Discussion on collision behavior of NS-3: https://groups.google.com/forum/#!topic/ns-3-users/Zx-I9DnXxaA https://groups.google.com/forum/#!searchin/ns-3-users/collision%7Csort:date/ns-3-users/40vBIJbic5E/zRFEhKWMAQAJ