**Abstract:**

Here in this programming project, we study and analyze the traditional/original backoff algorithm that the WiFi 802.11 MAC protocol uses for collision avoidance. We try to understand the purpose of such an algorithm and try to make it more efficient.

The backoff algorithm used by the WiFi MAC protocol was designed to decrease wastage of system time by avoiding collision and thus increase the overall throughput. Thus to do so all the devices in the network maintain their own contention window, that determines how much time will that device wait before retransmission once the collision happens. The contention window starts with size=2 and can go on to size=1024. On each collision, the colliding devices doubles the size of their contention window (with an attempt to reduce the chance of collision in future). The main disadvantage of this algorithm is devices keep waiting even when the medium is idle and it is possible that even after waiting, collision will happen. Hence, though the algorithm increases throughput, it is not the very efficient at doing so.

**EXPONENTIAL BACKOFF ALGORITHM**

Device has a packet x to transmit

If system is busy?

NO

YES

X selects a random waiting time from contention window and waits

Doubles Contention Window Size.

At CW=0, it transmits. Collision?

YES

NO

X transmits the packet successfully

Thus to overcome this limitations of the exponential (as the backoff window increases exponentially on collision) backoff algorithm, we implement an algorithm that is based on probability of successful transmission or failure that each device takes into consideration before placing a data packet on to the medium.

This algorithm aims at saving the time that is wasted in waiting for the contention window to drop down to zero before the device can attempt to transmit the data packet. This is because the contention window can size upto a large number like 1024 and if the system is idle, a lot of time is wasted by the device waiting for the backoff to exhaust.

Here in this project we implement two variations of this p-persistent backoff algorithm

1.) p-persistent backoff algorithm with fixed probability

2.) p-persistent backoff algorithm with variable probability

The main difference between these two types is that in the first type, the probability is kept constant for the entire algorithm where as in the second type, each device assumes that there are 'n' devices active at that moment and calculates probability such that n\*p < 1.

We will see later in our simulation statistics that having a variable probability is more accurate as compared to the fixed one as the number of devices in the connection in real life can vary.

**P-Persistent Backoff Algorithm**

Device x has a packet to transmit

If system is busy?

NO

YES

X defers transmission with probability (1-p). This p is calculated assuming number of active devices

The device transmits with probability p. Collision?

Device waits for random amount of time.

YES

NO

X transmits the packet successfully

**Simulation Program Details:**

As a part of this project, I have developed a ‘python’ based program that can compare the performance of the three kind of MAC algorithms described above. The program uses the extended ‘matplotlib’ tool to plot various parameters of the system like throughput, latency, collisions, etc. The most important parameter of this MAC protocol is the ‘throughput’ and we see how the probability based algorithms improve the throughput of the WiFi system. The figure below shows the comparison of the throughputs of various collision avoidance techniques, and we see that a p-persistent algorithm helps us achieve the maximum throughput (almost equal to 1).

(Reference: [www.slideshare.net](http://www.slideshare.net/))

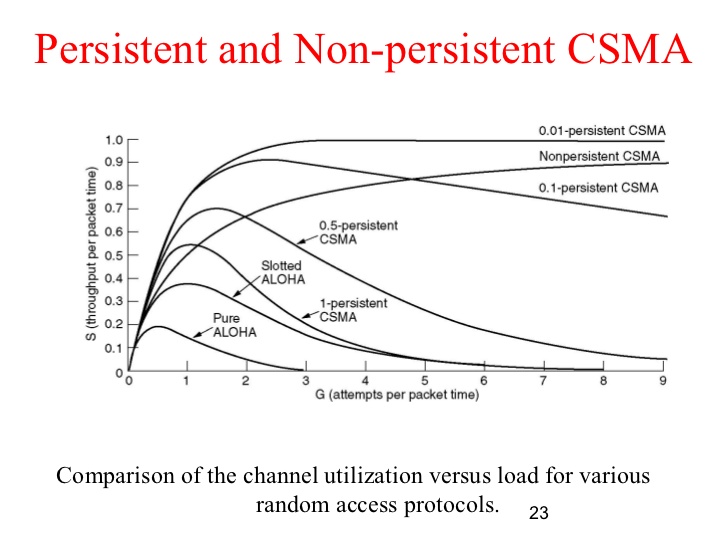


Fig 1.1

The graph plots the respective values of the load that the system is facing (G) versus the throughput of the system(S). In our system we were not able to plot this because of the program complexity was increasing. Hence, we plotted average values of each algorithm which can be calculated with as follows:

Throughput = load i

load + latency

**Inputs and Outputs:**

The simulation program takes various inputs from the user like number of devices in the system, size of data packets and how many data packets each device wish to transmit. There is not limitation on these inputs i.e. the user can simulate the program for any number of devices with any number of data packets.

In the course of simulation, the program computes system parameters based like total time of simulation (in microseconds), the throughput of each kind of algorithm , the latency and load, etc. which are presented as output in form of graphs.

**Assumptions and Limitations of our program:**

* The program assumes same packet size (in terms of time slots) for all devices that is taken as input from the user.
* The frames like DIFS, SIFS and ACK are assumed to fixed length in terms of slot times and remain static for the entire simulation program.
* The program does not take in to consideration the RTS/CTS problem currently but can be extended/upgraded in future to support it.
* The program does not plot the throughput/latency of the system in a timely manner as shown in figure 1.1, but shows the comparison that is average at the end of simulation. (This was tried but it increased program complexity and was not yielding correct output.)

**Code Explanation:**

The entire program is commented which makes easy to understand by anyone. The program is modularized into various routines which are given calls whenever required. The program uses various inbuilt functions like random() and time() and also makes use of outside tools like matplotlib to plot various bar graphs.

**Simulation statistics:**

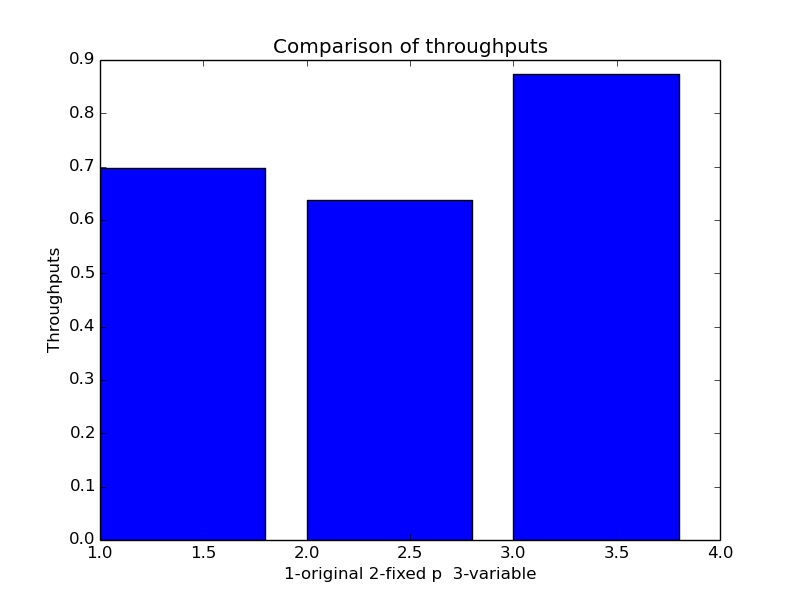
Simulation 1 -

(The probability of fixed p-persistent is taken as 0.2)

Number of devices = 6

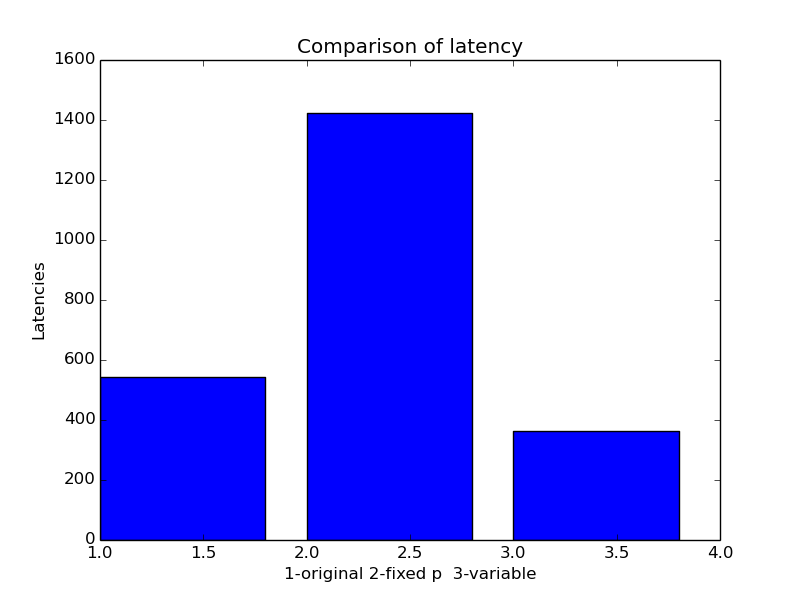
Length of each packet = 100

The following plots were captured from simulation 1 –

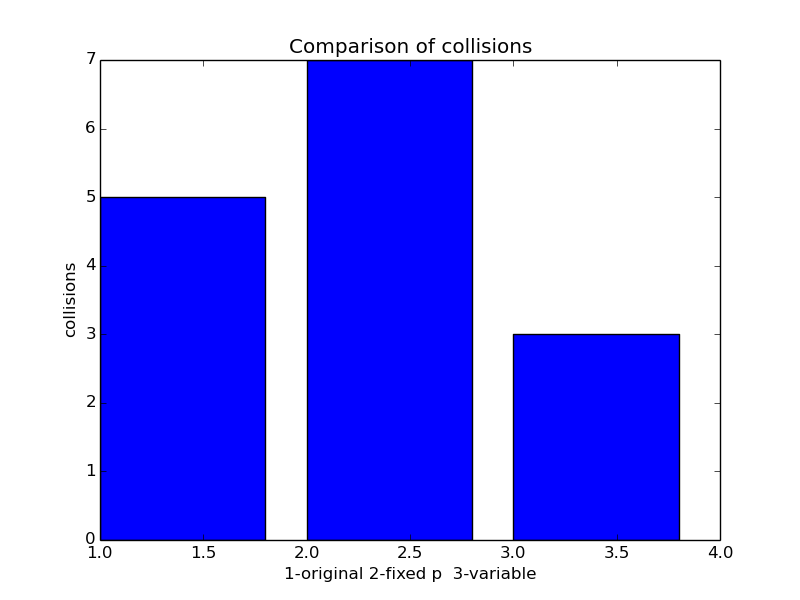


From this, its evident p-persistent is efficient as compared to original and fixed-p persistent backoff algorithm. The throughput of exponential backoff will not be this high in real life as the number of devices will be more and also we have taken scenario in which other packets of same device cannot transmit together.

The throughput of fixed probability persistent algorithm is based on the value of p (which we have taken as 0.2) and other factors like number of devices,etc. This can improve by assuming accurate value of p

Simulation – 1 continued :

As stated above, latency of fixed-p can be improved by guessing correct value of p. The 3rd one which is a variable – p persistent backoff algorithm has the lowest latency as expected.



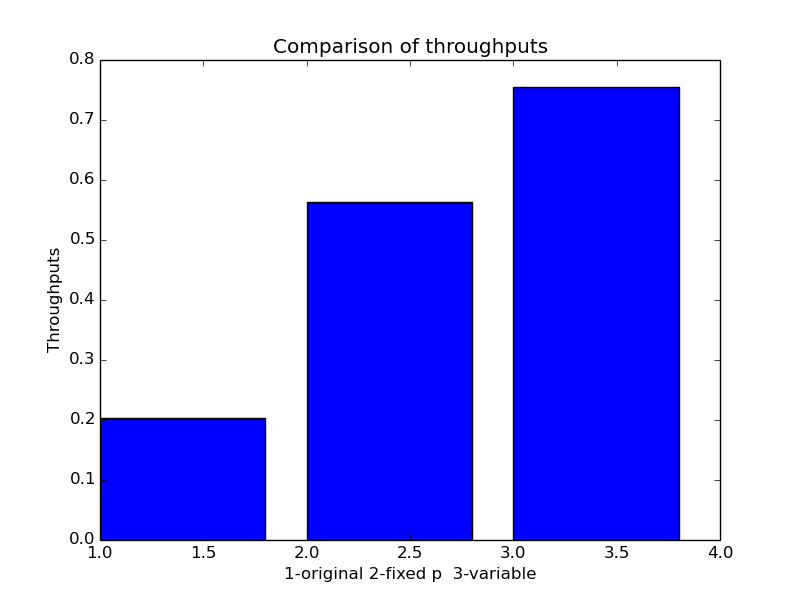
Simulation – 2

(The probability of fixed p-persistent is taken as 0.2)

Number of devices = 10

Length of each packet = 300 timeslots

The following plots were captured from simulation 1 –



Same as first simulation, we see that the throughput of variable – p persistent algorithm is most efficient out of all three. The maximum efficiency of variable – p persistent algorithm can go to around 0.95 or may be more.

