Windowed backoff algorithms for WiFi: theory and performance underbatched arrivals

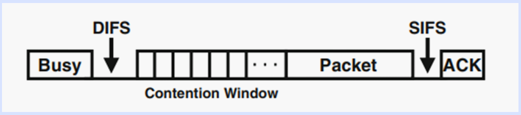
Saadi Saadi and Musa Eisa

Project Overview

In this project we want to see how well BEB works when a group of data packets all try to use a wireless channel at the same time. We tested it alongside newer methods, using a simulation tool called Network Simulator 3.
After that we want to present our new algorithm and compare it with the other algorithms using ns3.
ns-3 is a free open source project aiming to build a discrete-event network simulator targeted for simulation research and education.

BEB Overview

When a station wants to transmit data on a shared channel, it checks if the channel is idle. If it is, the station waits for a Distributed Inter-Frame Space (DIFS) duration and then sends the data. If the channel is busy due to another station transmitting, the first station waits until that transmission finishes. Afterward, it goes through DIFS again and then selects a random waiting time between 0 and w-1 slots, where w represents the contention window size. The contention window acts like a range of possible waiting times. Each station begins with a specific window size, and after each data transmission attempt, this window size can change. The size adjustment is based on the algorithm being utilized



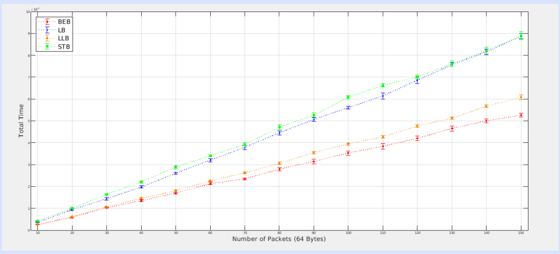
Experimental Setup

We examine a single batch of n packets that simultaneously begin their contention for the channel(n stations every station sends a single packet)

Parameter	Value
Slot duration	9 µs
SIFS	16 µs
DIFS	34 µs
ACK timeout	75 µs
Transport layer protocol	UDP
Packet overhead	64 bytes
Contention-window size min.	4
Contention-window size max.	4096
RTS/CTS	Off

following those assumptions we talked about earlier. We tested them with different numbers of packets: 10, 20, all the way up to 150 and one access point(AP). For each case, we did the test 30 times to get an average execution time and cw slots.

Simulation Results

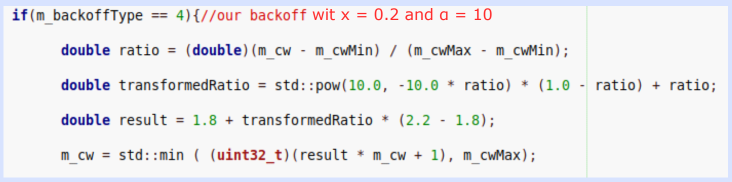


Conclusions

-For algorithm design, optimizing CW slots at the expense of increased collisions is a poor designchoice.
-Backing off slowly is bad.

Our Algorithm

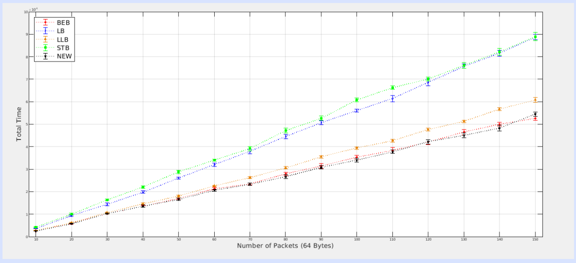
we came with this algorithm trying getting better performance than BEB
We define ratio to be:"current_cw – min_cw" /"max_cw – min_cw"
transformedRatio = 〖10〗 ^(−α∗$ratio$)∗(1 −$ratio$)+$ratio$
Factor = 2 – x + transformedRatio*2x
New_cw_size = factor * old_cw_size
This approach enables us to achieve the desired behavior based on the contention window size, while ensuring that the factor remains closely aligned with 2 (within the range of 2-x to 2 + x).
here is the code:



```
if(m_backoffType == 4){//our backoff wit x = 0.2 and α = 10

      double ratio = (double)(m_cw - m_cwMin) / (m_cwMax - m_cwMin);

      double transformedRatio = std::pow(10.0, -10.0 * ratio) * (1.0 - ratio) + ratio;

      double result = 1.8 + transformedRatio * (2.2 - 1.8);

      m_cw = std::min ( (uint32_t)(result * m_cw + 1), m_cwMax);
```

Simulations' Relsuts

Algorithm 3 with α = 10 and x = 0.2 (our algorithm is the black one)



Conclusions

Algorithm 3's performance is somewhat superior to BEB, though the difference is not substantial. Therefore, it's likely more practical to opt for BEB due to its much simpler implementation