Report: testing CSMA/ECA implementation

Luis Sanabria-Russo

Abstract

In order to avoid unnecesary repetitions of tests and in the hopes of finding clear answers to our questions, this report gathers results for CSMA/ECA tests made at using the reserved WiFi channel 14. Results highlight the steep difference between the number of retransmitted frames seen with our prototype to be later compared against what it is expected according to analytical models.

1 Introduction

This document is a work-in-progress, it does not matter when you read it.

In order the identify the different characteristics of the implementation of CSMA/ECA using OpenFWWF, we first sniff the traffic generated by CSMA/CA stations. Then, we implement CSMA/ECA and contrast the results.

1.1 The setup

As usual, we performed the test with an Access Point (AP), a wired server and the Alix 2d2 as a host. The access point broadcasts a network that has MAC address access restrictions to prevent devices not in the test from joining in.

The test environment is not as clean as we would like to be. Although we performed channel scans and set up the AP to the WiFi channel with less intruders, it did not prevent external transmissions from causing collisions in our experiment.

If not mentioned otherwise, there are always four stations transmitting UDP segments at a fixed rate of 1 Mbps to a wired iPerf server. All traffic from the stations is captured using Wireshark and processed later with a custom made parser [1]. At the moment, we use the retransmission flag and the transmission time of each frame to compute throughput, average time of arrival and number of retransmitted frames for each host.

2 OpenFWWF: modifying the `set_backoff_time` function

You may find the code of the backoff function in Listing 1.

```
// **********************************************************
// FUNCTION:   set_backoff_time
// PURPOSE: Updates backoff time for contention operation.
//
```

```
 5  set_backoff_time:;
 6    srx 2, 8, [SHM_TXFCUR], 0x000, GP_REG5;
 7    or  [0x162], 0x000, SPR_BASE4;
 8    and SPR_TSF_Random, CUR_CONTENTION_WIN, GP_REG5;
 9    je  CUR_CONTENTION_WIN, DEFAULT_MIN_CW, deterministic_backoff;
10    jext COND_TRUE, continue_backoff_calculations;
11  continue_backoff_calculations:;
12    or  GP_REG5, 0x000, [0x05,off4];
13    add [0x04,off4], GP_REG5, [0x06,off4];
14    or  [0x06,off4], 0x000, GP_REG5;
15    jnzx 0, 11, SPR_IFS_STAT, 0x000, need_more_time_to_elapse;
16    or  SPR_IFS_0x0e, 0x000, GP_REG1;
17    jl  GP_REG5, GP_REG1, need_more_time_to_elapse;
18    sub GP_REG5, GP_REG1, GP_REG0;
19    or  GP_REG0, 0x000, SPR_IFS_BKOFFDELAY;
20    jext  COND_TRUE, end_backoff_time_update;
21  deterministic_backoff:;
22    or  0x100, 0x000, GP_REG5;
23  need_more_time_to_elapse:;
24    or  GP_REG5, 0x000, SPR_IFS_BKOFFDELAY;
25  end_backoff_time_update:;
26    or  SPR_IFS_BKOFFDELAY, 0x000, [0x16F];
27    or  CUR_CONTENTION_WIN, 0x000, [0x03,off4];
28    ret lr1, lr1;
```

Listing 1: `set_backoff_time` function

The modifications we made based on our current understanding of the code are:

1. Line 9: we check if the current contention window is equal to the minimum contention window, if so it will *jump* to the `deterministic_backoff` branch. This is true when:
   - the machine powers on.
   - a packet is discarded due to reaching the retransmission limit.
   - After a successful transmission.
2. Line 10: if the current contention window is not the minimum, then the execution flow will jump to the `continue_backoff_calculations` branch.
3. Line 22: the deterministic backoff is stored in the General Purpose Register 5 (`GP_REG5`), which is the one used for the backoff operations prior storing its value in memory (in Line 19, this is the default operation, we did not add this line). In this example the value `0x100` is stored in `GP_REG5`.

Apart from the modifications specified above, in some scenarios we also changed the minimum contention window to be two times the tested deterministic backoff, that is, if the backoff is $B_d = 16$ slots, then the minimum contention window is set to $CW_{\min} = 32$, and so on.

3 Effects of the deterministic backoff

According to the simulations performed with CSMA/ECA, after a transitory phase nodes achieve a collision-free schedule due to the deterministic backoff after successful transmissions. Nevertheless, in the real implementation we see retransmitted frames at all stages of the test. Figure 2 shows the approximate number of retransmitted frames sent to the wired server while performing an `iperf` test.

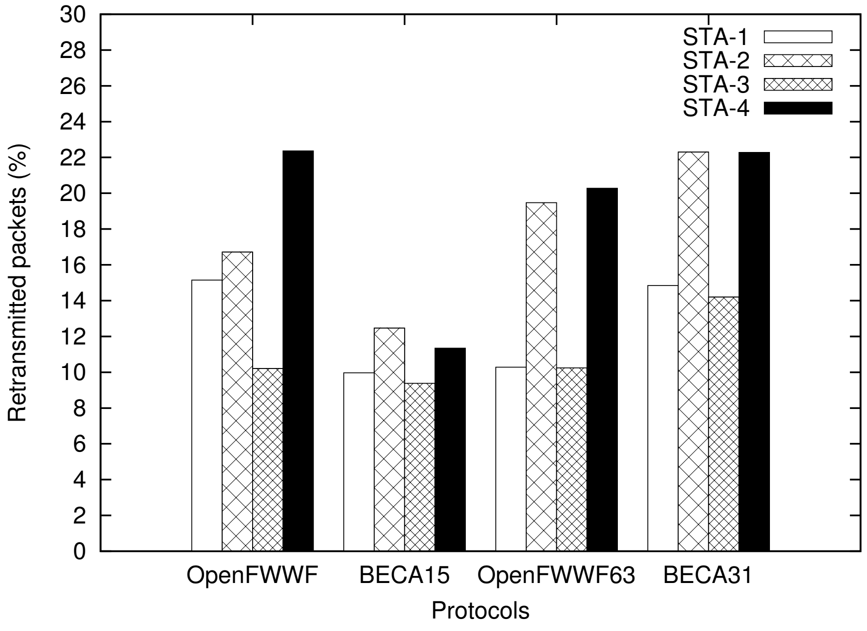

Fig. 1: Percentage of retransmitted frames

In the figure, when using CSMA/ECA the deterministic backoff is half the minimum contention window. That is, for BECA15: $B_d = 15$, $CW_{\min} = 31$ and so on. The number besides OpenFWWF determines the minimum contention window value; for OpenFWWF63: $CW_{\min} = 63$.

Figure 2 shows how the average number of retransmitted frames of BECA15 is lower than the average of OpenFWWF. One would expect to see a reduction of the fraction of retransmitted frames when doubling $CW_{\min}$ and B_d; nevertheless this does not happen.

After doing various tests, we have noticed that whenever the value of $CW_{\min}$ is changed, BECA starts behaving as DCF. The reason why this happens is still unknown.

3.1 Not changing $CW_{\min}$

To test our hypothesis, we repeated the tests changing only B_d and fixing $CW_{\min} = 31$, which is its default value. Figure **??** shows the average number of retransmissions for different values of B_d against OpenFWWF.

As B_d grows, the number of retransmitted packets is reduced given that some of the possible collisions are avoided due to the deterministic backoff. Two special cases, namely BECA127 and BECA255 experience a slight increase in the number of retransmitted packets. The reason is still unclear.

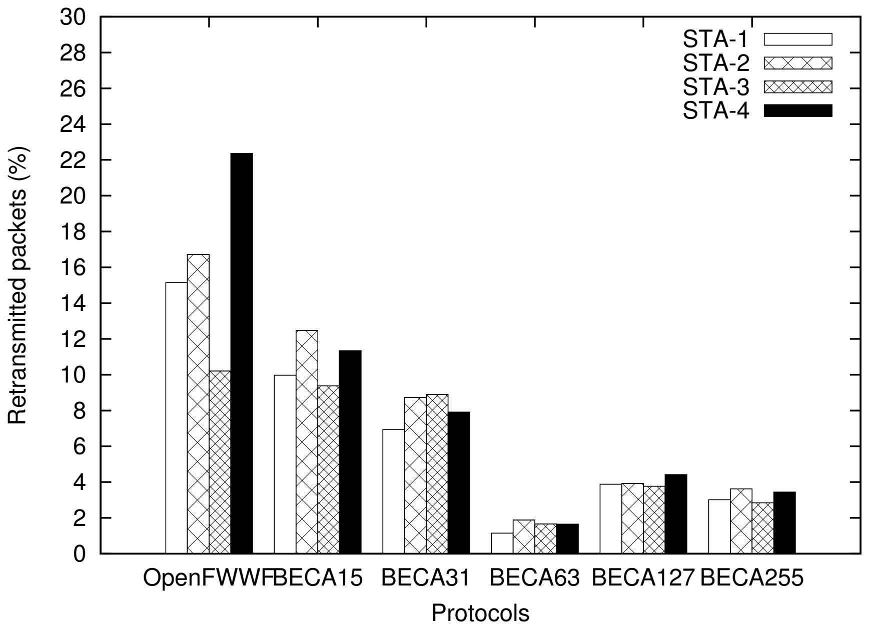

Fig. 2: Percentage of retransmitted frames fixing $CW_{\min} = 31$

3.1.1 Just two stations

In the eve of the INFOCOM demo, we decided also to perform the tests with only two stations (the demo will consist on comparing 2 stations running BECA against 2 stations runnning DCF). Figure 3 shows the results.

We can see how the average fraction of retransmitted packets per station is similar in both Figure 2 and 3 for the same protocol.

The average throughput per station achieved with each tested protocol is shown in Figure 4.

References

[1] L. Sanabria-Russo. (2013) Parsing CSV files from Wireshark. [Online]. Available: https://github.com/SanabriaRusso/sharktoolStatistics

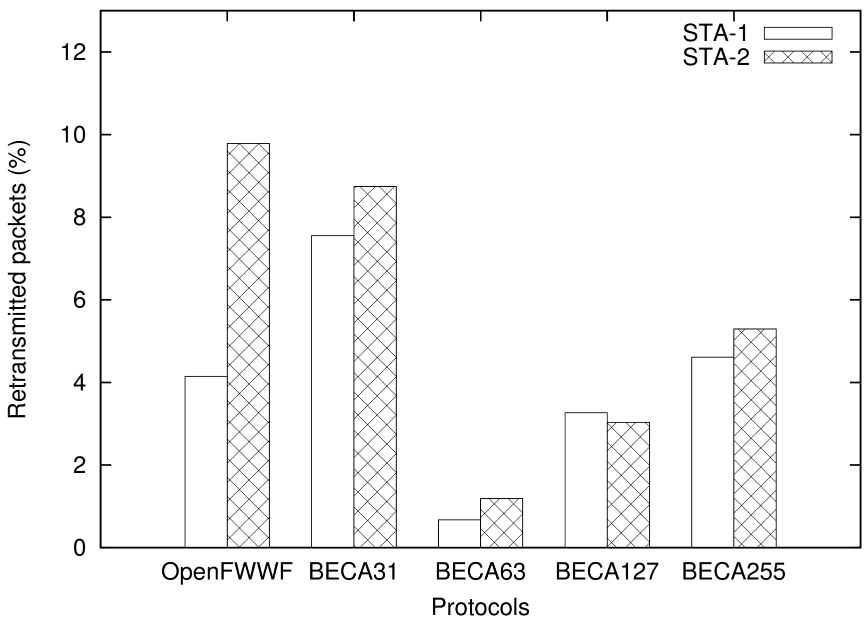

Fig. 3: Percentage of retransmitted frames fixing $CW_{\min} = 31$ and only two contenders

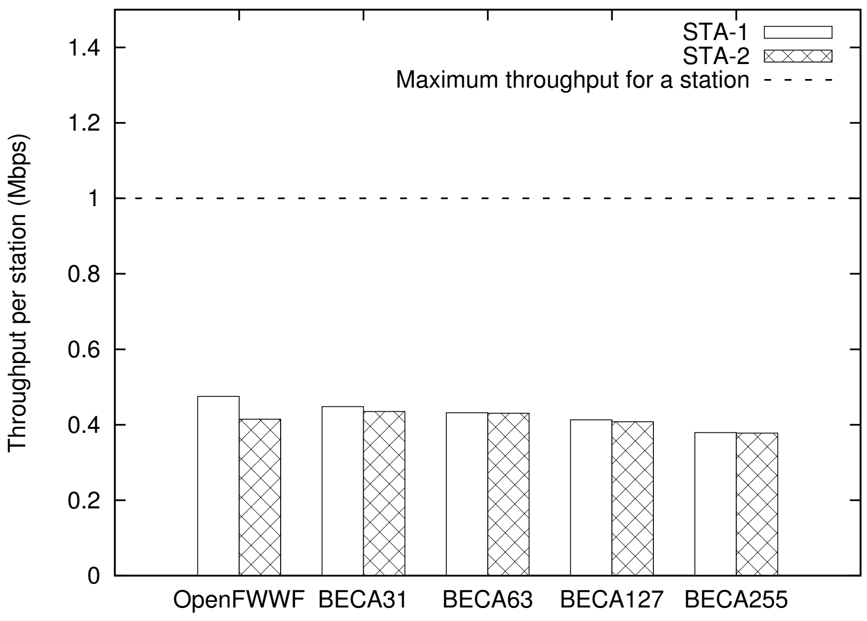

Fig. 4: Average throughput per station