Report: testing CSMA/ECA implementation

Luis Sanabria-Russo

Abstract

We have performed many tests with the CSMA/ECA prototype. Although we were unable to find a clean environment that would remove the possibilities of external noise, tests revealed the implementation of the deterministic backoff after a successful transmission, which reduced the number of retransmitted frames. This report aims at gathering all the implementation details in order to be to minimize errors while repeating the tests, and also help us understand the behaviour or the implementation.

1 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.
//
  set_backoff_time:;
    srx 2, 8, [SHM_TXFCUR], 0x000, GP_REG5;
    or  [0x162], 0x000, SPR_BASE4;
    and SPR_TSF_Random, CUR_CONTENTION_WIN, GP_REG5;
    je  CUR_CONTENTION_WIN, DEFAULT_MIN_CW, deterministic_backoff;
    jext COND_TRUE, continue_backoff_calculations;
  deterministic_backoff:;
    or  0x100, 0x000, GP_REG5;
  continue_backoff_calculations:;
    or  GP_REG5, 0x000, [0x05,off4];
    add [0x04,off4], GP_REG5, [0x06,off4];
    or  [0x06,off4], 0x000, GP_REG5;
    jnzx 0, 11, SPR_IFS_STAT, 0x000, need_more_time_to_elapse;
    or  SPR_IFS_0x0e, 0x000, GP_REG1;
    jl  GP_REG5, GP_REG1, need_more_time_to_elapse;
    sub GP_REG5, GP_REG1, GP_REG0;
    or  GP_REG0, 0x000, SPR_IFS_BKOFFDELAY;
    jext  COND_TRUE, end_backoff_time_update;
  need_more_time_to_elapse:;
    or  GP_REG5, 0x000, SPR_IFS_BKOFFDELAY;
  end_backoff_time_update:;
    or  SPR_IFS_BKOFFDELAY, 0x000, [0x16F];
    or  CUR_CONTENTION_WIN, 0x000, [0x03,off4];
    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 12: 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 21, 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, 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.

2 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.

3 File transfers using `scp`

We wanted to know approximately how long it takes CSMA/ECA stations to transmit 20 MB of data (see Table 1) to the wired server, this way we might identify faulty cards. Although the tool might not be appropriate for a complete analysis because of the involvement of higher layers congestion control (TCP), the Secure Copy (`scp`) command may serve as an initial pointer. It is not considered a reference metric, though. We are still looking for the appropriate tool.

Tab. 1: CSMA/ECA nodes transfering 20 MB of data with no other contenders

	Avg. Tx Time (s)			
(B_d, $CW_{\min}$)	**Node 1**	**Node 2**	**Node 3**	**Node 4**
(31, 63)	8	9	10.2	9.8

The data in Table 1 suggests that cards are able to perform similarly.

3.1 Two simultaneous contenders

CSMA/ECA's enhancements are related to a better collision avoidance, thus trying with only one station might in fact result in a lower throughput due to the more aggressive backoff mechanism in CSMA/CA.

Table 2 shows the average time it took to transfer 20 MB of data in a network with two CSMA/ECA contenders.

Tab. 2: Two CSMA/ECA nodes transfering 20 MB of data

	Avg. Tx Time (s)	
(B_d, $CW_{\min}$)	**Node 1**	**Node 2**
(255, 511)	17.8	17.6