Report: more efficient packet source

Luis Sanabria-Russo

***Abstract*—Since we started to use the modified version of the COSM simulator to implement CSMA/ECA$_{\text{Hys+FS}}$, we faced long simulation times to derive meaningful data. This is due to the different timers used to mimic a packet generator with Poisson arrivals. In this report we present a polished version of multiple CSMA/ECA$_{\text{Hys+FS}}$ instances in a node to provide traffic differentiation. Additionally, we further developed the packet source at each node to optimise simulation performance. Results show traffic differentiation for different access categories while greatly reducing the simulation time.**

I. Why is it important to modify the source?

In our attempt to show that CSMA/ECA$_{\text{Hys+FS}}$ is able to support many more contenders in a collision-free schedule, we need to test our approaches with an increased number of nodes. Further, to provide meaningful results each point in a figure should be representative. To achieve this, several long simulations are required which in turn take considerable *computation time* (around 30 seconds for a single run with 70 nodes).

The computation time is in its majority composed by timers. These timers dictate the generation of a new packet and follow a Poisson distribution. If a single node now is packed with four sources (following the Access Categories (AC) of EDCA), the computation time increases due to the added timers of each source.

If we are willing to simulate a great number of nodes with multiple ACs, we need to modify the packet source so it becomes computationally efficient.

The Fix

Instead of creating four sources (one for each AC), I implemented (what I called) a packet source router.

It is composed of a single source, a router and as many AC as needed (see Figure 1). The mechanism works as follows:

- The source generates packets according to a Poisson distribution with a user-defined λ. This will be called the overall *load*.
- The user also defines a load share for each AC. The load share is equivalent to the probability that a packet is destined to a determined AC, i.e.: $P(BE) = 35\%$ for Best-Effort, $P(BK) = 35\%$ for Background, $P(VI) = 20\%$ for Video and $P(VO) = 10\%$ for Voice.
- The router then redirects each generated packet to its corresponding AC based on the user-defined load shares.

This fix achieved a computation time reduction of around 80% on specific simulation scenarios.

II. Throughput

Figure 2 shows the aggregated throughput of each AC and the overall system throughput. The source settings are detailed in Table I.

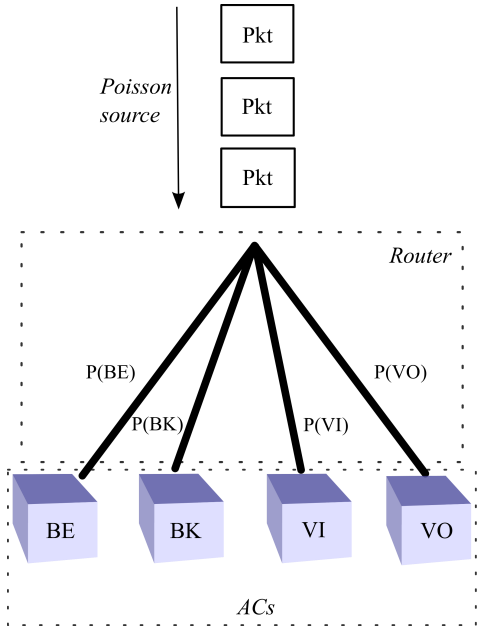

Fig. 1. Source diagram: generated packets are redirected by the router to each AC MAC queue according to load-shares.

TABLE I
Source parameters

Parameter	Value
λ	100 Mbps / 1024 B
BE share	35%
BK share	35%
VI share	20%
VO share	10%

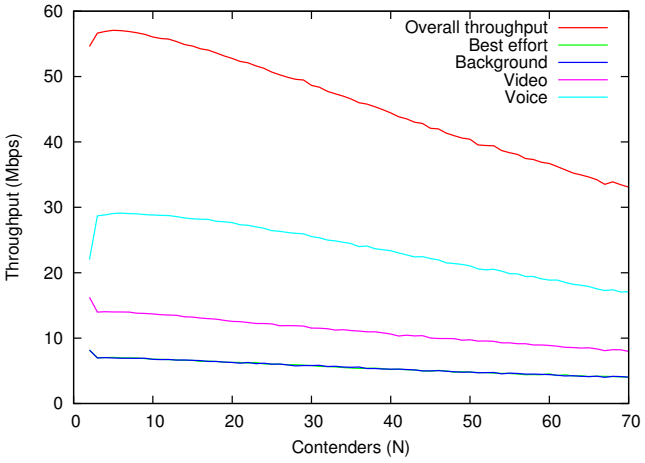

Fig. 2. Aggregated throughput per AC with the more efficient source.

III. Ongoing work

Now that we have multiple instances of CSMA/ECA$_{\text{Hys+FS}}$ in a single node working more efficiently, we are prepared to

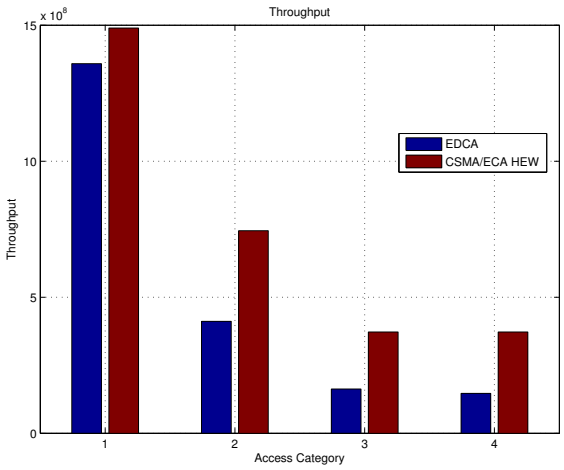

Fig. 3. Aggregated throughput per AC. CSMA/ECA HEW uses the new random backoff scheme.

introduce the modifications that would provide better collision avoidance. For instance:

1) A better selection of a random backoff: when an AC needs to pick a random backoff, it should also consider the backoff counters of the other ACs in the node. This would allow it to pick a backoff that will not cause an internal collision.
2) Implement AIFS so we can safely compare EDCA with $\text{CSMA/ECA}_{\text{Hys+FS}}$.

Point 1 has already been developed. It follows the condition that a new random backoff should not be the same as the current timers for the other ACs. Also, it should not collide with successful ACs that pick a deterministic backoff. Figure 3 and Figure 4 show the aggregated throughput for each AC and the internal collisions when using the our new random backoff scheme, respectively. The figures highlight an increase in the throughput due to the reduction on internal collisions.

The implementation of this new *pseudo-random* backoff can be found in [1].

REFERENCES

[1] Sanabria-Russo, L. (2014) Github repository: PseudoRandom-Backoff. Webpage, accessed January 2015. [Online]. Available: https://github.com/SanabriaRusso/PseudoRandomBackoff

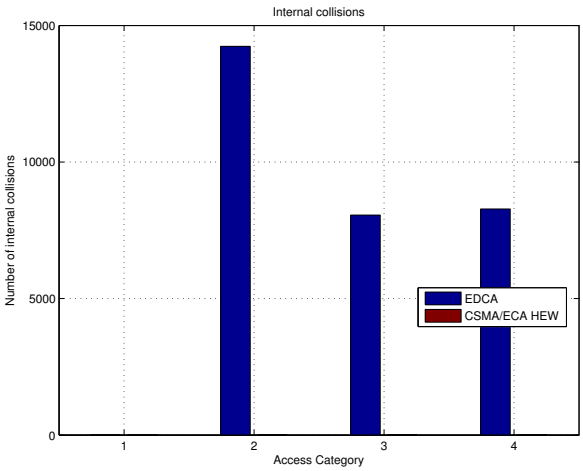

Fig. 4. Internal collisions per AC. CSMA/ECA HEW uses the new random backoff scheme.