Rob Hanlon (rob@cs), Daniel Otero (oterod@cs)

Group name: rob

CSE 461

Zahorjan

Homework #2: RFID Simulation

**Design for Iteration #1**

Our first implementation of a collision resolution algorithm was derived from that suggested by Annex D of the protocol spec. In this algorithm, the strawman algorithm in the assignment write-up is augmented with variation in slot size depending on tag responses. When no response is received in response to a reader’s query, it is assumed that the slot size used in that query was too large in relation to the number of tags in range of the reader. Therefore, we reduce the size of the window in order to increase the probability of a tag’s random slot counter being zero. On the other hand, if a collision is detected in response to a reader query, it is assumed that the slot size used was to small for the number of tags present, causing multiple tags to generate the same random slot counter value. In this case, we increase the slot size in order to reduce the probability of collisions. Decreases and increases in slot size are always exponential, and slot size is a power of two.

However, despite the reasonableness of this approach, it is possible to toggle back and forth unnecessarily between no response and collision. If the number of tags is sufficiently far from both 2Q-1 and 2Q (where Q is the number of bits for slot size), the reader might constantly change slot size up and down without ever drawing the desired tag response. Therefore, rather than increasing Q in increments of one each time, we add and subtract a positive constant C < 1, allowing us to “retry” a particular slot size a few times before giving up.

Lastly, in this initial approach, we used CRC error correction on all frame types, in a naïve attempt to maximize the number of detected, corrupted frames.

**Results from Iteration #1**

We evaluated the effectiveness of our first attempt at this inventorying algorithm by comparing our results to those of our classmates, using the collaborative graphing tool provided for this assignment. We noticed that our approach resulted in 100% precision, but had poor recall in comparison with other students’ results. Whereas a large majority of other students’ algorithms had near-perfect recalls for low numbers of tags, our recall ranged from ~50% for 100 or less tags to ~35% for 160 tags.

**Design for Iteration #2**

Our evaluation of iteration #1 led us to consider a number of alternatives aimed at increasing recall. Our first intuition, which proved to be unsatisfactory, was that even upon receiving a query response from a single RFID tag, our algorithm would blindly ignore corrupted RN16 and EPC frames. We therefore tried re-requesting several new RN16 frames, but found that this approach yielded only a modest increase in recall (~5%). This increase was not up to par with the large recall percentages that our classmates were able to get.

Our next modification was to relax the strictness of error detection. We realized that nothing that could happen to a Query frame could be harmful to the correctness of our algorithm. If some tags received Query frames with corrupted slot sizes, there might be a higher chance of collision, in which the only consequence would be a retry by the reader. If no tags responded as a result of an unrealistically large (and corrupted) slot size, this would again result in a retry. If the inventoried bit in the Query frame was corrupted (changed from false to true), there would be a chance that an already-inventoried tag would be considered again. Not only is this fairly rare (less than a (1% BER \* ) chance that an already-inventoried tag would move to the reply state), but the potentially undesired multi-inventorying of that tag could be avoided by simply enforcing uniqueness of EPCs in the reader. In the final case, despite some (or all) Query frames being corrupted, a single tag responds with an RN16, in which case all goes as planned despite corruption. Yet while there are no possible gains from using CRC on Query frames, throwing out corrupted Query frames actually hurts performance. Because CRC cannot improve correctness, all the algorithm does by throwing away corrupted Query frames is waste opportunities to make progress.

While using CRCs on Query frames proved to be detrimental, using CRCs on other frame types is crucial, as each of the other frame types store information that is much more delicate. EPC frames, for example, need reliable corruption detection in order to avoid inventorying a false positive.

**Results for Iteration #2**

As a result of our modifications, recall increased to 97% for 20 tags, while remaining the same at 160 tags. At 160 tags, the amount of time that it takes to successfully identify all tags far exceeds the window of time allowed.