6. Conclusion

At the start of this thesis the research question was clearly defined. The theoretical advances had been made, now it only needed the practical proof of concept. We implemented a working version of the Self-Stabilizing End-to-End Communication in Bounded Capacity, Omitting, Duplicating and non-FIFO Dynamic Networks algorithm in TinyOS, according to the demands of the original paper in which it described the necessary steps to be taken for the practical version to be completely working.

We completed the practical demands that were needed in order to fulfill the self-stabilization criteria. We created several versions of the algorithm, ran the simulations to define the optimization variables, and we met the demands of the self-stabilization criteria:

1. Insert
2. Duplicate
3. Reorder

Using LibReplay, we tested the algorithms and saw that they successfully passed the test.

The issues at hand concerning the development of such an algorithm in TinyOS are still relevant though. The debugging difficulties concerning sensor networks and distributed systems make it difficult to at times find a problem in the code. The TinyOS documentation lack severely, and since the abandonment of this operating system is slowly starting to become clear, it doesn't look like this is going to improve.

In light of the results we established, and the myriad of problems we faced, I look forward and take a moment to advice the future researcher on the possibilities that are still present in this work.

6.1 Future work

As mentioned in the beginning of this work; the algorithm was written as part of a research group called Gulliver who are doing research and activities concerning sensor networks. The eventual goal is for this algorithm to be used in a working environment, but for this is not yet ready. More research is necessary to understand where exactly in can deliver it's highest value.

Second, there were a few variables where I didn't had the time to invest a lot of effort into them during this project. One of them is the Reed-Solomon Error Corrrecting Code that I took from a library online. After going through the research, Reed-Solomon was en still is the perfect candidate for this algorithm as an Error Corrrecting Code. The only problem is the high complexity; it would

take a separate proect to develop an RS ECC algorithm. But taking it from a online library, because of time constraints, meant I couldn't dive in the code and see if I could remove some parts or tweak some elements that could maybe improve the perfomance even more.

Finally, looking back at the many problems I faced during development, and the painstakingly slow process of making applications in TinyOS, I would strongly advise any future researcher to switch to more supported operating systems like ContikiOS. TinyOS has it's place, but it is losing a race it can't win anymore. I base my statements on the founder and main developer of TinyOS, Professor Philip Levis of Stanford University. A few years ago he wrote an extended article about his experiences during ten years of TinyOS development. Looking back and looking at the current status of TinyOS, even he sees the logical trend of research mobing their aim away from TinyOS because of the non-active development.