Distributed Consensus for low-power multi-hop networks in `contiki-ng`

Alberto Spina

October 29, 2018

Objective

This is a two-step project.

1. **Implement Glossy**. Glossy[1] is an efficient network flooding and time synchronization protocol which has been written by Academics for Contiki 2 and Contiki 3. My aim is to port Glossy to `contiki-ng`[9]. Glossy will expose a `send_when` primitive, which will allow to send constructively-interfering messages across the network (i.e. not necessarily flood messages).
2. **Network-wide Consensus**. Based on a working implementation of Glossy I aim to implement A^2 Synchotron[2], a system that brings distributed consensus to low-power multi-hop networks, within the `contiki-ng` Operating System.

Motivation

This project brings together two fields I am incredibly passionate about: WSNs (via Contiki), Software Engineering and Consensus.

- **Consensus**. Reaching agreement within a distributed system is always challenging, especially within WSNs where faults are likely to occur, communication over radio is lossy and processes can crash. As outlined within the A^2 paper[2], the proposed algorithm builds on top of 2PC and 3PC consensus protocols, allowing for network-wide agreement with different consistency/liveliness tradeoffs.
- **Contiki**. With the new `contiki-ng` OS we are seeing a complete new iteration of the Contiki Operating System. With a cleaned up codebase and a lot of room for extensions, `contiki-ng` is setting itself as a standard for IoT programming. Over the years a multitude of protocols were developed by Academia for the various Contiki 2.x and 3.x Operating Systems; it is now time that ground-breaking papers such as Glossy receive a working and tested implementation within upstream Contiki.
- **Software Engineering**. By developing within `contiki-ng` one is pushing into an actively developed IoT kernel. This means that all code must be proven to be tested, cross-platform and reliable. Code will have to be tested within simulations (handled via Cooja), real world test-beds, and individual components must be unit tested within the `contiki-ng` CI pipeline.

Since its publication in 2011, Glossy has had a lot of attention from Academia. Other than A^2 Consensus, there are a number of other projects which are based on top of Glossy and would require an implementation to be pushed into `contiki-ng`.

- **LWB**: Low-Power Wireless Bus[3].
- **Splash**: Fast Data Dissemination with Constructive Interference in Wireless Sensor Networks[4].

- **Choco**: Low-Power, End-to-End Reliable Collection Using Glossy for Wireless Sensor Networks[5].
- **Sparkle**: Exploiting Concurrent Transmissions for Energy Efficient, Reliable, Ultra-Low Latency Communication in Wireless Control Networks[6].

Approach

This project will consist of a multi-step process:

- Step 1: There are a number of custom-tailored Glossy implementations for Contiki 2.x and 3.x. This will be my starting point. I will initially be looking at available implementations adapting them to ensure compilation for MSP430 architecture. Testing will occur within Cooja and on real motes. I will then deploy a working implementation onto available test-beds to check I am still able to obtain the results outlined on the paper.
- Step 2: I will then port Glossy to `contiki-ng`. This will require the addition of tests, thorough code rewriting, restructuring and refactoring. This code will then also be tested on testbeds to see any performance impacts of the new kernel (w.r.t. the results reported on the paper). A new `send_when` primitive will be exposed which, supported by Glossy, allows lossy links to become reliable.
- Step 3: Given a working implementation of Glossy I will implement A^2 consensus. A^2 Synchotron[11] has publicly available implementations which will be used as a starting point. It is, though, provided as a standalone kernel forked off from Contiki 3.

Challenges and Issues to be resolved

There are a number of potential challenges involved with this project:

- Real world hardware. Implementations will always be tested on real hardware; there will likely be failures, and other hardware technical problems which might slow down progress.
- Cryptic papers. Even though implementations of certain papers are available, this is Academic code used as a Proof of Concept. Porting to `contiki-ng` will mean completely rewriting certain sections thoroughly understanding papers published in the past few years.
- Testbeds. There are a number of testbeds available to research groups around the globe. I will request access to these in order to test on hundreds of nodes with varying topology. There is likely high demand for these testbeds and they might not be available with a lot of support. Glossy and A^2 Synchotron have been tested on FIT-IoT[7] and FlockLab[8] testbeds.

References

[1] Federico Ferrari, Marco Zimmerling, Lothar Thiele, and Olga Saukh. 2011. Efficient Network Flooding and Time Synchronization with Glossy. *In Proceedings of the Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN).*

[2] Beshr Al Nahas, Simon Duquennoy, and Olaf Landsiedel. 2017. Networkwide Consensus Utilizing the Capture Effect in Low-power Wireless Networks. *In Proceedings of SenSys 17, Delft, Netherlands, November 68, 2017, 14 pages.*

[3] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele. 2012. Low-Power Wireless Bus. *In Proceedings of the Conference on Embedded Networked Sensor Systems (ACM SenSys).*

[4] Manjunath Doddavenkatappa, Mun Choon Chan, and Ben Leong. 2013. Splash: Fast Data Dissemination with Constructive Interference in Wireless Sensor Networks. *In Proceedings of the Symposium on Networked Systems Design Implementation (USENIX NSDI).*

[5] M. Suzuki, Y. Yamashita, and H. Morikawa. 2013. Low-Power, End-to-End Reliable Collection Using Glossy for Wireless Sensor Networks. *In IEEE 77th Vehicular Technology Conference (VTC Spring).*

[6] Dingwen Yuan, Michael Riecker, and Matthias Hollick. 2014. Making Glossy Networks Sparkle: Exploiting Concurrent Transmissions for Energy Efficient, Reliable, Ultra-Low Latency Communication in Wireless Control Networks. *In Proceedings of the European Conference on Wireless Sensor Networks (EWSN).*

[7] Cedric Adjih, Emmanuel Baccelli, Eric Fleury, Gaetan Harter, Nathalie Mitton, Thomas Noel, Roger Pissard-Gibollet, Frederic Saint-Marcel, Guillaume Schreiner, Julien Vandaele, and others. 2015. FIT IoT-LAB: A large scale open experimental IoT testbed. *In IEEE 2nd World Forum on Internet of Things (WF-IoT).*

[8] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel. 2013. FlockLab: A Testbed for Distributed, Synchronized Tracing and Profiling of Wireless Embedded Systems. *In Proceedings of the Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN)*

[9] Contiki-NG: *The OS for Next Generation IoT Devices.* [ONLINE]: https://github.com/contiki-ng/contiki-ng

[10] csarkar. *Tailored- LWB.* [ONLINE]: https://github.com/csarkar/tailored-lwb

[11] iot-chalmers. *A2-Synchrotron.* [ONLINE]: https://github.com/iot-chalmers/a2-synchrotron