Verification of the Contiki-NG Operating System for the Internet of Things

CS 485 – Undergraduate Research

Supervisor: Frédéric Loulergue

Fall 2019 or/and Spring 2020

1 Context

Connected devices and services, also referred to as Internet of Things (IoT), are gaining wider and wider adoption in many security critical domains. This raises important security challenges, which can be addressed using formal verification.

Contiki [5] is a popular open-source operating system for IoT devices providing full low-power IPv6 connectivity, including 6TiSCH, 6LoWPAN, RPL, or CoAP standards. It is implemented in C with an emphasis on memory and power optimization, and contains a kernel linked to platform-specific drivers at compile-time. When Contiki was created in 2002, no particular attention was paid to security. Later, communication security was integrated.

Formal verification of the Contiki code was not performed until very recent case studies [8, 9, 2, 7, 3, 4]. These case studies were performed using the ACSL specification language [1] and the deductive verification plugin WP of Frama-C [6].

Contiki-NG [1] is a new version of Contiki.

2 The Projects

The following sub-sections describe several possible projects.

2.1 Ring Buffer

The goal of the project is to specify and verify, using Frama-C and WP, the ring buffer library of Contiki-NG. This module implements a circular buffer where bytes can be read and written independently. The buffer itself is an array.

[1] `http://www.contiki-ng.org`

2.2 Stack & Queue

The goal of the project is to verify, using Frama-C and WP, the stack and queue modules of Contiki-NG. Both modules rely on the list API of Contiki-NG. A first step is therefore to adapt if necessary the specifications of the Contiki list API to Contiki-NG. There are currently three different specifications of the list API module. In this project, you will experiment with all three and compare the pros and cons of each approach for the verification of the queue and stack modules.

2.3 Double Linked Lists

The goal of the project is to verify, using Frama-C and WP, the double linked lists module of Contiki-NG. Single linked lists have already been verified in two different ways [2, 3]. In this project you will consider the adaptation of these two approaches to the verification of double linked list and compare the approaches.

2.4 Circular Lists

The goal of the project is to verify, using Frama-C and WP, the circular lists module of Contiki-NG. Non circular lists have already been verified in two different ways [2, 3]. The verification of circular lists raises new challenges. Part of the work will be to determine if one of these approaches can be adapted to circular lists, or if a completely new approach is required. For this project CS451 is a prerequisite.

2.5 Heap Memory

A simple memory module was provided by Contiki. This module `memb` was specified and verified using Frama-C [8]. The `heapmem` module of Contiki-NG is a dynamic heap memory allocator similar to malloc/free in standard C. The goal of this project is to specify and verify `heapmem` using Frama-C and its plugin WP. For this project CS451 is a prerequisite.

3 Requirements

Minimum Requirements

- Good knowledge of programming in C
- A taste for formal reasoning
- CS 396
- Enrolled in CS 451 for Fall 2019

Preferred Requirements

- CS 451

4 Laboratory

Students will be given access to SSERL[2] and its machines including the Titan workstation (256 Gb of memory and 32 cores).

References

[1] Patrick Baudin, Jean C. Filliâtre, Thierry Hubert, Claude Marché, Benjamin Monate, Yannick Moy, and Virgile Prevosto. *ACSL: ANSI/ISO C Specification Language*, February 2011. `http://frama-c.cea.fr/acsl.html`.

[2] Allan Blanchard, Nikolai Kosmatov, and Frédéric Loulergue. Ghosts for Lists: A Critical Module of Contiki Verified in Frama-C. In *Nasa Formal Methods*, number 10811 in LNCS, pages 37–53. Springer, 2018. doi:10.1007/978-3-319-77935-5_3.

[3] Allan Blanchard, Nikolai Kosmatov, and Frédéric Loulergue. Logic against ghosts: Comparison of two proof approaches for a list module. In *ACM Symposium on Applied Computing (SAC)*, pages 2186–2195. ACM, 2019. doi:10.1145/3297280.3297495. Best Paper Award.

[4] Allan Blanchard, Frédéric Loulergue, and Nikolai Kosmatov. Towards Full Proof Automation in Frama-C using Auto-Active Verification. In *Nasa Formal Methods*, LNCS. Springer, 2019. to appear.

[5] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. Contiki – a lightweight and flexible operating system for tiny networked sensors. In *LCN 2014*. IEEE, 2004.

[6] Florent Kirchner, Nikolai Kosmatov, Virgile Prevosto, Julien Signoles, and Boris Yakobowski. Frama-C: A software analysis perspective. *Formal Asp. Comput.*, 27(3): 573–609, 2015. doi:10.1007/s00165-014-0326-7.

[7] Frédéric Loulergue, Allan Blanchard, and Nikolai Kosmatov. Ghosts for lists: from axiomatic to executable specifications. In *Tests and Proofs (TAP)*, volume 10889 of *LNCS*, pages 177–184. Springer, 2018. doi:10.1007/978-3-319-92994-1_11.

[8] Frédéric Mangano, Simon Duquennoy, and Nikolai Kosmatov. A memory allocation module of Contiki formally verified with Frama-C. A case study. In *CRiSIS 2016*, volume 10158 of *LNCS*. Springer, 2016. doi:10.1007/978-3-319-54876-0_9.

[9] Alexandre Peyrard, Simon Duquennoy, Nikolai Kosmatov, and Shahid Raza. Towards formal verification of Contiki: Analysis of the AES–CCM* modules with Frama-C. In *RED-IoT 2018, co-located with EWSN 2018*. ACM, 2018.

[2] `https://sserl.github.io`