Des listes et leurs fantômes : vérification d'un module critique de Contiki avec FRAMA-C

Allan Blanchard, Nikolai Kosmatov, Frédéric Loulergue

Des listes et leurs fantômes : vérification d'un module critique de Contiki avec FRAMA-C* (résumé étendu)

Allan Blanchard[†1], Nikolai Kosmatov[‡2], and Frédéric Loulergue[§3]

[1]Inria Lille – Nord Europe, Villeneuve d'Ascq, France
[2]CEA, LIST, Software Reliability Lab, 91191 Gif-suf-Yvette, France
[3]School of Informatics, Computing, and Cyber Systems, Northern Arizona University, Flagstaff, USA

Les appareils et services connectés, aussi appelés internet des objets (*Internet of Things : IoT*) sont de plus en plus populaires dans des domaines où la sécurité est critique. Ces questions de sécurité sont une cible de choix pour l'usage de méthodes formelles.

Ce travail se concentre sur Contiki, un système d'exploitation *open-source* populaire pour les objets connectés, qui propose notamment une pile IPv6 pour environnements contraints. Il est implémenté en C avec une attention particulière pour l'optimisation de la consommation mémoire et énergétique, et est disponible pour de nombreuses plateformes physiques. Au début du développement de Contiki en 2002, la sécurité n'était pas un objectif central. La sécurité des communications a été ajoutée plus tard, mais la vérification formelle du code n'a commencé que très récemment.

Nous présentons la vérification formelle du module de listes chaînées de Contiki. Ce module est critique pour le système d'exploitation, il est utilisé par 32 modules de Contiki et appelé plus de 250 fois dans la partie cœur du système. Son API est riche, il est possible d'ajouter ou enlever des éléments à n'importe quelle position dans la liste. Elle garantit l'unicité des éléments au sein des listes : avant toute insertion, le module s'assure de supprimer l'élément de la liste avant de l'ajouter à la position voulue. Finalement, Contiki ne permet pas l'allocation dynamique, contrairement aux implémentations classiques des listes, le module de Contiki ne fait donc pas de telle opération.

La vérification formelle du module est effectuée grâce au langage de spécification ACSL et au greffon WP de la plate-forme FRAMA-C. Notre approche se base sur des tableaux fantômes (*ghost*) utilisés comme représentation des listes chaînées

*Cette soumission est un résume étendu de l'article [1] accepté à NFM 2018.

[†]allan.blanchard@inria.fr
[‡]nikolai.kosmatov@cea.fr
[§]frederic.loulergue@nau.edu

manipulées. Cette représentation permet de raisonner à propos des éléments de la liste automatiquement mais demande de maintenir une relation forte entre les listes et leurs tableaux compagnons.

La relation entre un tableau fantôme et une liste chaînée est formulée à l'aide d'un prédicat inductif qui à partir d'un $index$ donné du tableau, et parallèlement à partir de l'adresse du premier élément de la liste, assure que l'on trouve l'adresse de chaque cellule i à la position $index + i - 1$ du tableau. Maintenir la relation entre la liste et le tableau fantôme nécessite de mettre à jour le tableau chaque fois que la liste est mise à jour, ce qui est réalisé à l'aide de fonctions fantômes.

Cette relation étant inductive, elle ne peut pas être manipulée directement par les prouveurs SMT qui ne sont pas capables de réaliser des preuves par induction. Cette formalisation inductive est donc accompagnée d'un ensemble de lemmes qui expriment des relations utiles à partir du prédicat inductif, et qui peuvent être directement manipulés en preuve automatique.

Ces lemmes permettent notamment de couper une liste et la plage du tableau auquel elle est liée en plusieurs sous-listes associées à des sous-plages du tableau (et inversement de fusionner des sous-listes en une liste complète). Ces relations sont par exemple nécessaires dans le cas d'une suppression d'un élément : on doit d'abord prouver que l'on peut couper la liste en deux sous-listes séparées par l'élément à supprimer, puis prouver que l'on peut « recoller » les deux sous-listes sans l'élément à supprimer.

A partir du code original du module (176 lignes de C), nous avons produit environ 1400 lignes d'annotations, dont 500 pour les contrats et 240 pour les définitions et lemmes. Sur les 798 obligations de preuves générées par WP, 24 correspondent aux lemmes et sont prouvées à l'aide de l'assistant de preuve COQ, par induction sur le prédicat de liaison. 770 obligations sont prouvées automatiquement. Sur les 4 restantes, 2 sont prouvées avec COQ et 2 avec le prouveur interactif de WP (TIP).

En travaux futurs, nous prévoyons d'effectuer la vérification de modules client du module de listes. Pour permettre l'utilisation du test, nous avons reformulé nos spécifications pour les rendre exécutables. Par ailleurs, nous prévoyons de comparer la formalisation actuelle avec des formalisations plus abstraites : listes logiques et fonctions partielles contiguës.

Remerciements. Ce travail a été partiellement soutenu par le CPER DATA et le projet VESSEDIA, financé par le programme européen pour la recherche et l'innovation Horizon 2020 selon la convention de subvention No 731453. Les auteurs remercient également l'équipe FRAMA-C pour les outils et le support, ainsi que Patrick Baudin, François Bobot et Loïc Correnson pour leurs discussions et conseils.

Références

[1] A. Blanchard, N. Kosmatov, and F. Loulergue. Ghosts for Lists :A Critical Module of Contiki verified in FRAMA-C. In *NASA Formal Methods Symposium (NFM)*, LNCS, Newport News, VA, USA, April 2018. Springer.