Science of Computer Programming ••• (••••) •••–•••

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Design and implementation of a cross-layer IoT protocol

Davi Resner [*], Gustavo Medeiros de Araujo, Antônio Augusto Fröhlich

Software/Hardware Integration Lab, Federal University of Santa Catarina, PO Box 476, 88040-900, Florianópolis, SC, Brazil

A R T I C L E   I N F O

Article history:
Received 1 February 2017
Received in revised form 14 August 2017
Accepted 17 August 2017
Available online xxxx

Keywords:
Cross-layer design
Communication protocols
Internet of Things
Wireless Sensor Networks

A B S T R A C T

Cross-layer communication protocols can significantly improve several aspects of wireless networks, particularly energy consumption and bandwidth. However, they break with the traditional layered architecture that has been so successful for over 30 years. A fully fledged cross-layer stack requires a sophisticated software design to be maintainable and reusable. The Trustful Space-Time Protocol (TSTP) is a cross-layer protocol designed to deliver authenticated, encrypted, timed, and georeferenced messages containing data compliant with the International System of Units (SI) in a resource-efficient way. By integrating shared data from multiple networking services into a single communication infrastructure, TSTP is able to eliminate replication of information across services, achieving small overhead regarding control messages. The complexity of TSTP's features, its broad range – from the application to the Medium Access Control, – and its experimental nature bring diverse requirements beyond those usually considered in most software designs. In this work, we show how we avoided a monolithic implementation of the cross-layer approach with a component-based design, exploring template metaprogramming techniques to adapt and combine basic building blocks. An event-driven architecture that makes use of zero-copy buffers and metadata is used to handle crosscutting concerns. We validate the proposed design with an implementation for IEEE 802.15.4 networks and a set of OMNet++ simulations for relevant scenarios, showing that we achieve a light TSTP implementation that is successfully able to save energy and bandwidth while keeping a delivery ratio close to 100%.

1. Introduction

Wireless Sensor Networks (WSNs) have been the focus of intense research for well over a decade by now. Several physical layers have been proposed, along with a myriad of medium access and routing protocols [11] [19]. Such protocols have been made energy-aware [17]; aggregation and fusion strategies have been employed [14]; basic infrastructures have been enriched with location [20], timing [26], and security protocols [10]; operating systems have been designed to support higher-level abstractions [3], along with large-scale management systems built to handle the produced data properly [12]. We are currently seeing these networks being connected to the Internet of Things (IoT).

Extensive research is also being carried out on cross-layer optimizations for wireless communication [8] and WSN [16] protocols. These works focus on taking into account related information given by one layer to efficiently make decisions at a different layer in the communication stack. A cross-layer protocol design can eliminate a large number of control

[*] Corresponding author.
E-mail addresses: davir@lisha.ufsc.br (D. Resner), gustavo@lisha.ufsc.br (G. Medeiros de Araujo), guto@lisha.ufsc.br (A.A. Fröhlich).

http://dx.doi.org/10.1016/j.scico.2017.08.008
0167-6423/© 2017 Elsevier B.V. All rights reserved.

messages and improve decision making within the protocol by pig-tailing control information on ordinary data packets and subsequently organizing and sharing that information with all protocol components.

The main advantage of the traditional layered model is the inherent ability to induce highly cohesive, loosely coupled modular software implementations since layers see each other as black-boxes with a well-defined and small interface. Conversely, a cross-layer protocol calls for close component interactions, challenging designers aiming for traditional software engineering quality metrics such as modularity, reusability, and maintainability. Furthermore, embedded IoT devices usually have low processing power and very limited energy budgets, so the network protocol stack, which is often a major part of the whole system's functionality, must be efficiently implemented.

In this work, we present the component-based design of the cross-layer *Trustful Space-Time Protocol* (TSTP) [23], highlighting the close interactions among MAC, router, location, and time synchronization components. We discuss the techniques that were used in this design on the *Embedded Parallel Operating System* (EPOS) [13] to demonstrate that the tight relationships in the cross-layered approach can be modeled and implemented avoiding a monolithic, tightly-coupled software. We use template metaprogramming techniques to implement an event-driven architecture that moves packets stored in zero-copy, metadata-enriched buffers across protocol components efficiently at the same time as it eliminates unnecessary dependencies. We present a set of simulations that corroborate the benefits of the cross-layer approach regarding energy consumption and bandwidth when compared to other non-cross-layered protocols. We also evaluate the component-based implementation concerning size and performance to demonstrate the small overhead incurred by the chosen techniques, which is also confirmed by comparisons with the same traditional protocols used in the simulations. EPOS and TSTP are written in C++ and are freely available online at https://epos.lisha.ufsc.br/. The EPOSMote hardware used to automate UFSC's Smart Solar Building, the real IoT platform used for the evaluations in this paper, is a free hardware project available at the same site.

We present the general cross-layer design of TSTP in Section 2, with a focus on the MAC and routing interaction, which encompasses much of the challenges that arise in the cross-layer design. Section 3 explains the techniques used to achieve a light and modular implementation of that design. In Section 4 we evaluate the protocol's design and implementation by using simulations and measurements on a real IoT device. Related work is presented in Section 5, and Section 6 concludes the work.

2. Cross-layer protocol design

Cross-layer designs have been shown to be very efficient in optimizing wireless networks [27]. In practice, cross-layer designs usually work by taking information from one or more layers of a typical layered stack to optimize a set of parameters or make a decision in another layer or set of layers [8]. From the myriad of cross-layer proposals, a minority involves the application layer, and even fewer encompass the complete stack to present a truly application-oriented, domain-specific solution.

The *Trustful Space-Time Protocol* (TSTP) [23] is an application-oriented, cross-layer protocol for *Cyber-Physical Systems* (CPS) on a WSN or the IoT. Instead of focusing on keeping the original protocol interfaces in a modular, layered architecture with shared data, TSTP focuses on efficiently providing functionality recurrently needed by such systems: trusted, timed, geo-referenced, SI-compliant data that is resource-efficiently delivered to a sink or gateway. TSTP grants these functionalities directly to the application in the form of a complete communication solution, which allows the design of optimized, synergistic cooperation of sub-protocols while eliminating the need for additional heterogeneous software layers that come with an integration cost and often result in replication of data.

TSTP integrates time synchronization, spatial localization, security, MAC, routing, and a data-centric API. In previous work [23], we evaluated the whole design analytically and showed that the cross-layer integration of those services yields, in some scenarios, a node deployment 31% faster than the deployment time achieved by a design with all these services operating individually, as in a traditional layered stack. Here we extend the analysis to the integration, implementation, and interaction of the components, particularly the two most impacting ones (MAC and routing), and enrich it with simulations using the Castalia [2] framework on the OMNeT++ [18] simulator.

2.1. Messages, space and time

Fig. 1 shows the format of the header present in every TSTP message. It consists mostly of information necessary to characterize the *data* being transmitted: *where* (`Origin(x,y,z)`) and *when* (`Origin(t)`) it was produced, and *until when* (`Expiry`) it should be consumed. Other fields pertain coordinates scaling, message relay, or serve to keep nodes synchronized in space and time. TSTP messages are signed and their payloads are encrypted [24], so no explicit error-detecting codes are used (corrupted messages will fail decryption).

TSTP synchronizes nodes in time with high accuracy ($\approx$1 μs for IEEE 802.15.4) using the *Speculative Precision Time Protocol* [25]. If the speculative technique fails because the network is too silent at a moment and the estimated clock skew reaches half of the maximum tolerated value, then a message with the `Time Request` bit is sent to neighbors (1 hop), which react replying with an empty message (containing only the header) just to fulfill the requirements of the time synchronization algorithm. Nodes are synchronized in space using a speculative *Heuristic Cooperative Calibration Positioning System* (HeCoPs) [20]. Each message carries the last hop coordinates (`Last Hop(x,y,z)`), that node's confidence on its