TINET+TECS: Component-based TCP/IP Protocol Stack for Embedded Systems

Takuro Yamamoto*, Takuma Hara†, Takuya Ishikawa†, Hiroshi Oyama‡, Hiroaki Takada† and Takuya Azumi*
*Graduate School of Engineering Science, Osaka University
†Graduate School of Information Science, Nagoya University
‡OKUMA Corporation

***Abstract*—Embedded systems are applied to Internet of Things (IoT), and the high productivity of embedded network software is required. TINET is a TCP/IP protocol stack for embedded systems. Although TINET is a compact TCP/IP protocol stack, it consists of many complex source codes. Therefore, it is difficult to maintain, extend, and analysis the software. To improve the scalability and configurability, this paper has proposed TINET+TECS, a component-based TCP/IP protocol stack for embedded systems: TINET componentized with TOPPERS embedded component system (TECS). The component-based TINET provides software developers high productivity such as change of network buffer and adding/removing TCP (or UDP) function. We evaluate the component-based TINET compared with the original TINET. We confirm that the overheads of execution time and memory consumption are low, and that the configurability is improved.**

I. Introduction

Internet of Things (IoT) is an essential keyword for the next era [1] [2]. Various things, such as wearable devices, smart devices, and smart homes, connected to the Internet will enrich our lives. Embedded systems are the elements constituting IoT, e.g., sensing data and controlling actuators. It is not practical to implement the same TCP/IP protocol stack as a general computer because embedded systems have several restrictions such as low memory capacity.

TINET (Tomakomai InterNETworking) is a compact TCP/IP protocol stack for embedded systems [3]. TINET supports the ability such as minimum copy frequency and elimination of dynamic memory control. TINET needs only small memory for its TCP/IP protocol stack; therefore it is suitable for embedded systems. However, there are several issues that TINET consists of many complex source codes. In other words, TINET is composed of many files and defines many macros. This may take a lot of time for software developers to maintain, extend, and analysis the software. Embedded network software is required for the high productivity and quality.

An approach to improve software productivity is component-based development, which is a design technique that can be applied to reusable software development for embedded systems [4] [5], such as TECS [6] [7], AUTOSAR [8], and SaveCCM [9]. Component-based systems are flexible to software extension and specification changes. In addition, individual component diagrams enable the visualization of an entire system.

This paper proposes component-based TCP/IP protocol stack, i.e., TINET+TECS, to improve the configurability and scalability of software. TECS (TOPPERS Embedded Component System) [6] is utilized to componentize TINET, because TECS is a component system suitable for embedded systems. TECS supports static configuration, which statically defines component behaviors and interconnections, thus TECS can optimize the overhead of componentization.

In addition, the proposed framework utilizes dynamic connection, a ~~novel~~ method of TECS, to switch component bindings dynamically. A general TCP/IP protocol server dynamically processes the requested port i.e., HTTP (port: 80), HTTPS (prot: 443). However, embedded systems can not dynamically generate components due to the strict memory restriction. TINET+TECS statically generates components, and dynamically combines them. Therefore,

In the proposed framework, software application can be developed by not only TECS method but also existing method. Software appication can be developed as a TECS component because TINET+TECS is a component-based framework using TECS. Moreover, TECS supports an adapter to call functions of TECS components from non TECS codes. The legacy codes such as an existing application can be supported in the proposed framework.

This paper evaluates the overheads of execution time and memory consumption and the amount of code line change for adding/removing the functionalities, which demonstrates TINET+TECS can improve the configurability with small overheads.

Contributions: This paper provides the following contributions.

1) **Improve configurability**
   Since TINET+TECS is a component-based system, the software is flexible to change the configuration such as resizing network buffer, adding/removing TCP (or UDP) functions, and supporting both IPv4 and IPv6. In addition, a component diagram provides visualization of TINET, a complicated system.
2) **Dynamic connection method**
   Dynamically switching the binding of components, that is switching between a communication endpoint and reception point of TINET, realizes a TCP/IP protocol stack for embedded systems.

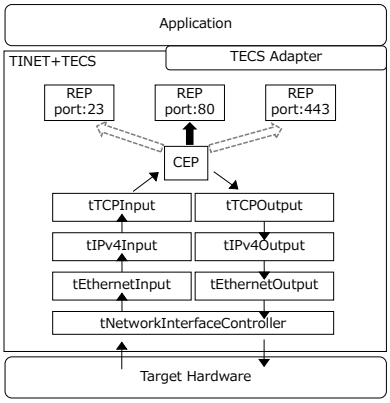

Fig. 1. System model

3) **Support legacy codes**
TINET+TECS can be applied to an existing application because TECS supports the adapter to call TECS functions from C codes.

Organization: The remainder of this paper is organized as follows. Section II introduces the system model and basic technologies, i.e., TINET and TECS. Section III describes the design and implementation of the proposed framework. Section **??** evaluates the proposed framework. Related work is discussed in Section V. Conclusions and suggestions for future work are presented in Section VI.

II. System Model

This section describes system model, including the basic technologies such as TINET and TECS. System model of the proposed framework is shown as Fig. 1. TINET+TECS is a component-based TCP/IP protocol stack, so TCP output task (tTCPOutput) and Ethernet input task (tEhternetInput) are implemented as TECS components. CEP and REP (Section II-A), which are also implemented as TECS components, dynamically switch bindings by TECS method. Moreover, TECS adapter supports the legacy codes such as an existing application.

A. TINET

TINET is a compact TCP/IP protocol stack for embedded systems based on the ITRON[1] TCP/IP API Specification [10], developed by TOPPERS (Toyohashi OPen Platform for Embedded Real-time Systems) Project [11]. TINET has been released as open source.

To satisfy restrictions for embedded systems such as memory capacity, size, and power consumption, TINET supports following functions:

- Minimum copy frequency
- Elimination of dynamic memory control
- Asynchronous interface
- Error detailed per API

[1]ITRON is a realtime operating system (RTOS) developed by TRON project.

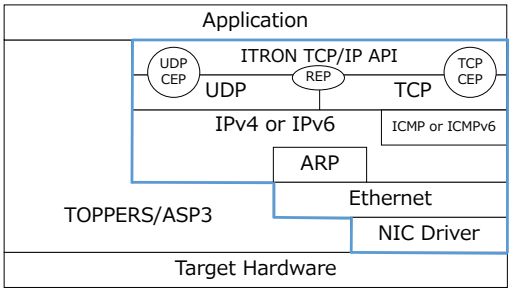

Fig. 2. Hierarchy diagram of TINET and TOPPERS/ASP3

1) Overview: TINET runs as middleware on TOPPERS/ASP3 [12] [13], which is a realtime kernel based on μITRON [14]. TINET also supports other RTOSs such as TOPPERS/ASP and TOPPERS/JSP because TINET is compatible with TOPPERS RTOS. Fig. 2 shows the hierarchy diagram of TINET and TOPPERS/ASP3. Users transmit and receive the data using a Communication End Point (CEP) which is an interface like a socket. In transmission process, headers are attached to the data body passed to the CEP at each protocol layer, and the data is transmitted from the network device. In reception process, headers of the data body received in the network device are analyzed at each protocol layer, and the data is passed to the CEP.

A TCP reception point called Reception Point (REP) is prepared to wait for a connection request from the partner side. An REP has an IP address (*myaddr*) and a port number (*myportno*) as attributes, and performs functions like *bind()* and *listen()*.

In TINET, the number of the data copy between each protocol layers is minimized. A TCP/IP protocol stack for general computers has large overheads of execution time and memory consumption because the data is copied at each protocol layers. To solve the problem, TINET does pass the pointer of the data buffer between each protocol layers, not perform data copy.

B. TECS

TECS is a component system suitable for embedded systems. TECS can increase productivity and reduce development costs owing to improved reusability of software components. TECS also provides component diagrams, which help developers visualize the overall structure of a system.

In TECS, component deployment and composition are performed statically. Consequently, connecting components does not incur significant overhead and memory requirements can be reduced. TECS can be implemented in C, and demonstrates various feature such as source level portability and fine-grained components.

1) Component Model: Fig. 3 shows a component diagram. A *cell*, which is an instance of a component in TECS, consists of *entry* ports, *call* ports, attributes and internal variables. An *entry* port is an interface that provides functions to other *cell*s, and a *call* port is an interface that enables the use of other *cell*'s