# Communication Protocol Decomposition and Component-based Protocol Submodule[1]

Tianzhou Chen, Quan Gan, Wei Hu, Jinhui Yu

College of Computer Science, Zhejiang University, Hangzhou, P.R.CHINA, 310027
{tzchen, whopawho, ehu, jhyu}@zju.edu.cn

**Abstract.** Due to evolving network technologies as well as increased and varying demands of modern applications on embedded systems, general-purpose protocol stacks are not always adequate. We examine the usefulness of component-based software engineering for the implementation of software communication systems. We present an architecture that allows dividing protocol software into fully de-coupled components. By plugging required components together, that is, loading proper protocol submodules for communication services, we rapidly prototype flexible, robust, and application-tailored communication protocols. The primary goal is the configurability of communication services, that is, the configurability of protocol submodules for each communication service according to application requirements and network capabilities. The achieved communication system supports efficiently and coordinately the multiple communication services customized by the coexisting applications.

The task of communication systems is to provide communication services for applications that exist in the system. We abstract the implementation of communication service customized by given application to a communication subsystem. A communication subsystem upgrades a given basic communication service in order to provide a target service requested by the application. However, the design and analysis of its architecture will be the most untrivial task, since the properties of this communication scheme will severely impact the performance of the whole system. In order to handle this, we have concentrated on the communication mechanisms, primitives and abstractions. To allow reuse, selection and integration of different modules, the communication and the computation will be separated and encapsulated. The communication subsystem is modeled as an ensemble of communicating modules. Each module is defined by its interface and its content where the interface is composed of a set of ports on which external or internal operations can be performed. This abstraction is reached when the module is represented by the protocol component and the communication is seen as the combination of requests and services. The protocol components serve as basic communication functions. They encapsulate the implementations of the protocol mechanisms, and provide the services externally by interfaces of them. The communication model is meant to manifest a unique interpretation of associations between components as a set of fundamental communication primitives. Moreover, these components may request other ones for services by interfaces, or their contents can be

[1] This work is supported by the national 863 high technology project and HP Embedded Laboratory of Zhejiang University.

composed of other component instances. Hence the communication subsystem is a stacking of all the required components, which are plugged together to realize a given communication service.

Due to various communication devices and application requirements the configurations of customized communication subsystems may be distinct. Therefore it is important to structure communication systems in a way that different subsystems can be supported and coordinated efficiently. We can employ different protocol components to serve the communication request that exists in different subsystems accordingly. Moreover, we support run-time reconfiguration of the communication subsystem. We examine the efficiency of the subsystem at run-time. When it does not adapt to the request of the application, more suitable component will be loaded to replace the one that does not match. This enables communication subsystems to adapt dynamically to changes in application requirements (e.g., switching from unreliable to reliable data delivery), communication system resources (e.g., buffer space and CPU load), and network characteristics (e.g., network congestion and routing). The replacement of protocol component is transparent to applications or the other components that refer to it as long as the interface remains the same. Proper communication components are dynamically loaded, that is, they are loaded into communication subsystem without rebooting the device. Communication subsystem adaptivity is important since applications and networks are dynamic entities that are not necessarily served most effectively by statically configured mechanisms.

The protocol component serves as the element of the communication system. It implements the basic communication service. These components are captured elegantly through the decomposition of the protocols. Protocol functions can often be provided by different algorithms (called mechanisms). Different mechanisms of one protocol function are implemented by the corresponding components while the interface remains the same. We give the name submodules for these correlative components. The novel approach to protocol component presented herein is that we can implement application- and device-customized submodules for the same communication function. New execution properties extending an existing service can be added by writing additional submodules and including them in the existing component suite.

This work is deployed on the platform of Linux, hence the protocol components are just implemented by Linux modules, which can be loaded into the kernel or removed dynamically without rebuilding the kernel. Our novel approach is to remove the original TCP/IP protocol stack from the Linux kernel. However, we load the corresponding modules that implement the communication components into the kernel after it starts up. These components will now provide the services that are provided originally by the TCP/IP functions.

There is approximately 1% performance lost compared to the monolithic implementation after the component-based mechanism is introduced, yet it is for the expense of encapsulation. However, the communication mechanisms can be tailored according to the device types and the applications running on them, which result in various protocol components that feature quite different performance characters. Thus we devise customized communication system with quite an increase in performance compared to the monolithic implementations.