The Rise of Protocol Engineering

FU CHUN JOSEPH LIN, AT&T Bell Laboratories
MING T. LIU, The Ohio State University

Communication software is a special class of software deployed in telecommunication or computer-communication networks to provide a set of well-defined services. Because communication software is distributed over a network's nodes, it requires built-in protocols to govern the communication among the network's components.

Protocols achieve desired services by coordinating and synchronizing the cooperation of distributed components, which are often implemented as processes. The services that protocols must accomplish are so complex that protocols are often organized as layers: Services are realized gradually, layer by layer.

Although most protocols used in communication networks have been standardized and specified in documents such as those published by the Comité Consultatif International Télégraphique et Téléphonique and the International Standards Organization, designing and testing the software that implements those protocols is difficult and error-prone, for several reasons.

- Many standards still rely more on informal text descriptions than on formal descriptions. This creates ambiguities and incompleteness in protocol specifications that must be resolved in the communication software's design.
- Even when standard protocol specifications are written formally, they may not have been validated sufficiently.
- The standards don't include machine-dependent protocol details, leaving gaps the software designer must fill in.

To tackle the problem, various disciplines from a number of well-established fields in computer science are combining to forge a new discipline called protocol engineering. The most notable of these fields are formal methods, software-engineering methodology, and knowledge-based engineering. Protocol engineering seeks to ease the development of communication software by providing engineering solutions to the problem of developing reliable protocols.

IN THIS ISSUE

This issue includes four articles about designing, validating, and testing communication protocols.

In the first article, "Protocol Design: Redefining The State of The Art," Gerard Holzmann identifies the major problem with traditional protocol development as the failure to rigorously verify the
Some well-established software fields are combining to forge the new discipline of protocol engineering, which seeks to ease the development of today's communication software.

protocol's design against its requirements before implementation. This results in the costly practice of detecting design errors in testing or even after field deployment.

To avoid this, Holzmann points out the need for a formal, unambiguous design notation and a technique that validates the design's correctness against requirements. He looks at the latest development in both formal specifications and validation techniques.

In the second article, "Protocol Validation for Large-Scale Application," we examine how to tackle the complexity of large-scale protocol validation effectively. An effective strategy has three ingredients: models, decomposition and abstraction, and reachability analysis. Using published examples and our industry experience, we explain each ingredient and reveal potential applications and research.

The third article, "Testing Communication Protocols" by Kshirasagar Naik and Behcet Sarikaya, describes the activities in protocol conformance testing, emphasizing test generation and verification.

To verify test cases, the authors translate both the formal protocol specification, written in Lotos, and the formal test specification, written in TTDN, into a common extended finite-state machine notation.

Finally, in "Using Artificial Intelligence in Communication System Design," Norio Shiratori and colleagues present their ideas for a totally integrated communication-system design environment using artificial intelligence. Its core is a set of cooperative expert systems that map the knowledge incorporated in high-level requirements to a low-level implementation.

Their integrated environment is the culmination of years of study in design-support systems. The authors describe in detail an expert system for protocol synthesis and a support system for protocol and communication-software design.

REFERENCES