Correctness Preserving Transformations
for Network Protocol Compilers

Tommy M. McGuire and Mohamed G. Gouda

Department of Computer Sciences

The University of Texas at Austin

October 31, 2001

Abstract

Strong abstractions provide the best basis for designing network protocols, but are difficult or inefficient to implement. We claim that a network protocol specified based on strong abstractions can be transformed into an implementation based on weaker abstractions which preserves the properties of the original specification. Further, this transformation can be done automatically and will produce efficient implementations. An open question is the possible interaction between the strong and weak abstractions and the performance of the system.

The problem of protocol design

The best way to design a network protocol is by basing the specification on strong abstractions. These strong abstractions make the protocol simple to understand and explain. Strong abstractions, however, are difficult and inefficient to implement.

For example, one useful abstraction[5] for protocol design is high-level, complex actions that are atomic across the whole protocol—an action can be executed in one process without interacting with other actions in the same or different processes. This abstraction makes design easier, by avoiding questions of simultaneity and synchronization between different actions. High-level atomicity also reduces the number of states of the protocol that need to be examined to ensure the correctness of the protocol. Unfortunately, this abstraction is very difficult to implement, since it requires global synchronization, and can lead to inefficient implementations.

High-level global atomicity can be weakened to local atomicity within a process with additional restrictions on inter-process communication. Fortunately, these restrictions are simply those of message-passing protocols in general: all inter-process communication consists of messages sent between the processes. As a result, the weaker abstraction is much easier to implement. Unfortunately, the weakened atomicity abstraction severely impairs the analysis and understanding of the specification, by greatly increasing the number of states that the protocol can be in. The weaker assumption is as inappropriate for protocol design as the stronger assumption is for protocol implementation.

The goal of our research[7] into correctness preserving transformations is to determine which strong abstractions can be weakened or modified and which need to be kept intact in order to preserve the properties of the protocol. Our research is similar to the work on parallelizing compilers for numerical analysis[3]. In that case, the algorithm is originally written for sequential execution and then automatically transformed to a form suitable for parallel execution. The parallelizing compiler determines which dependencies can be weakened while preserving the result of the computation.

Our approach is especially important to secure[6] and real-time[2] protocols. Secure protocols require formal analysis in order to provide confidence in the system in which the protocol plays a part; less rigorous approaches do not provide sufficient assurance. Real-time protocols have stringent requirements, which can be understood best by using an abstract model; without such a model it is difficult to even express the requirements.

Protocol development

While it is true that for any system, a small change in a specification can cause significant changes in behavior, both in terms of correctness and in terms of performance, this is more apparent in the network protocol domain. Network protocols need to preserve the end-to-end properties of the system as well as tolerate partial failure and avoid deadlocks while not adversely interacting with other protocols using the same communication medium. Under these circumstances, the effects of a change on the behavior of a system are rarely apparent. The correctness of a network protocol can be evaluated from a suitable formal specification but the performance in general cannot be, due to outside factors such as the interaction with other protocols in the environment.

As a result, experience with the protocol both in simulations and in running environments is an important part of the protocol development process. A cyclic approach is needed, beginning with the specification of a correct protocol, continuing with the transformation of that specification into an implementation and finishing with with the feedback from the implementation into the specification. A rapid turnaround of this cycle from specification to implementation is desirable. However, this rapid turnaround has proven difficult to achieve. Since protocol evaluation can be done in parallel, especially by simulations, the limiting factors are the specification and transformation to an implementation.

We believe that strong abstractions significantly ease and improve protocol specification. In order to support the cyclic protocol design process, though, the transformations which map a specification based on strong assumptions to an implementation based on weaker assumptions should be mechanical, embodied in a protocol compiler. This compiler, when given a protocol specification that satisfies a set of properties according to a set of strong assumptions, will produce an implementation which satisfies those same properties and which is based on weaker assumptions.

One remaining question about the automatic transformation of protocols is if there is a trade off in the protocol's performance based on the assumptions which are kept or modified. Implementation efficiency is an important goal for the transformation from a protocol specification, and there is no obvious reason why the performance of a transformed protocol would be objectionable[1, 4]. However, there may be an interaction between the performance of the protocol and the restrictions made by the transformation which may call for alterations of those restrictions, of the transformation process, of the properties of the protocol, and potentially for the stronger assumptions on which the protocol design is based.

One important point is that, even if the performance of a protocol implementation is unacceptable and no alterations are acceptable, the specification of the protocol based on strong assumptions is still valid and the optimization of the protocol, by hand if necessary, is made much easier by the existing, working implementation, even though it is inefficient.

Conclusion

The success of our approach would allow us to use strong abstractions to leverage protocol designer skill into more, higher-quality systems. The strong abstractions will increase the productivity of the designer as well as making the design of correct protocols easier while at the same time the transformation process will produce high-performance, correct implementations which can be used themselves or be the basis of further optimizations. By easing the task of designing and implementing protocols, this work will allow more systems to take advantage of more complex protocols, such as self-stabilizing and peer-to-peer protocols which have better failure handling and scaling properties.

References

[1] M.B. Abbott and L.L. Peterson. A Language-Based Approach to Protocol Implementation. *IEEE/ACM Transactions on Networking*, 1(1), 1993.

[2] Audio-Video Transport Working Group, H. Schulzrinne, S. Casner, R. Fredreck, and V. Jacobson. RTP: A Transport Protocol for Real-time Applications. RFC 1889, 1996.

[3] U. Banerjee, R. Eigenmann, A. Nicolau, and D.A. Padua. Automatic program parallelization. *Proceedings of the IEEE*, 81(2), 1993.

[4] C. Castelluccia, W. Dabbous, and S. O'Malley. Generating Efficient Protocol Code from an Abstract Specification. *Proceedings of the ACM SIGCOMM 1996 Conference*, 1996.

[5] M.G. Gouda. *Elements of Network Protocol Design.* John Wiley & Sons, 1998.

[6] M.G. Gouda, M. El-Nozahy, C.-T. Huang, and T.M. McGuire. Hop Integrity in Computer Networks. *Proceedings of the 8th IEEE International Conference on Networking Protocols*, 2000.

[7] T.M. McGuire. Correct Implementation of Network Protocols from Abstract Specifications. Dissertation proposal, The University of Texas at Austin, 2000. `http://www.cs.utexas.edu/users/mcguire/research/`.