SPECIFICATION, COMPOSITION, AND AUTOMATED
VERIFICATION OF LAYERED COMMUNICATION
PROTOCOLS

A Dissertation
Presented to the Faculty of the Graduate School
of Cornell University
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

by
David A. Karr
January 1997
SPECIFICATION, COMPOSITION, AND AUTOMATED VERIFICATION OF LAYERED COMMUNICATION PROTOCOLS

David A. Karr, Ph.D.
Cornell University 1997

Horus is a general-purpose layered message-passing system for distributed programming. A programmer of a distributed application can select protocol layers from among those provided by Horus and arrange these in a stack, thereby creating a custom-built message-passing protocol with strong (or not so strong) properties underneath the application.

For the full value of Horus's modularity to be exploited, an application programmer must be able to choose just the layers and stacking order that will provide the desired properties. A programmer who is limited to only a few "tried-and-true" alternatives may end up paying a performance cost (such as excessive synchronization messages) for unnecessary properties, simply because he or she cannot confidently build a less costly stack underneath a given application.

This dissertation describes a formal method that supports the engineering of new Horus protocol stacks by precisely specifying and mechanically verifying communication properties of these stacks. Various communication properties can be
described in English, but are also described succinctly in a mathematical model (the Temporal Logic of Actions) that supports sound reasoning about whether the properties are satisfied by an implementation. Each protocol layer guarantees various properties at its interfaces, depending on what assumed properties its neighbors provide to it. Relatively straightforward formal reasoning can then show that certain properties will be provided to the applications at the top of the stack.

This method of reasoning about protocol stacks can efficiently be automated so that it can be used by practitioners. A prototype of the verifier has been implemented in Java and published on the World Wide Web.
Biographical Sketch

David A. Karr was born in 1961 in the City of New York, about half a block from the apartment on 110th Street in Manhattan where he lived for the next eighteen years of his life, most of that time sharing a bedroom with his younger brother Robert. David’s father Ronald, a mechanical engineer, and his mother Elsa, a nurse, kept the apartment full of books (but devoid of television sets) and helped nourish David’s attraction toward academic studies. David also spent many enjoyable summers on his grandparents’ small farm in Great Barrington, Massachusetts.

David was fortunate to be instructed by many excellent teachers in the local public schools—P.S. 165 and Booker T. Washington Junior High School—before being admitted by examination to the Bronx High School of Science. Bronx Science provided many wonderful opportunities for David, including his first chance to program a computer: in 1976, David was writing programs in BASIC on a mechanical teletype attached to a Hewlett-Packard model 2000E. (Later, in another setting, he had the even greater pleasure of punching his own FORTRAN code on cards and handing the deck to a computer operator through the computer room...
David enjoyed all his academic subjects, graduating as valedictorian of his class in June 1979.

David attended Harvard College in Cambridge, Massachusetts, where, after a period of confusion because of his widely varied academic interests, he concentrated in mathematics. He supported his education in part by working as an undergraduate tutor and grader during term time, and by subcontracting in his father’s consulting practice at home, where he got his first taste of real engineering practices. He took a voluntary leave of absence for one semester, during which, among other things, he participated in political fund-raising and carried petitions from door to door. He completed his studies in January, 1984, accepted election to the Harvard chapter of Phi Beta Kappa, and graduated with honors that June.

From March, 1984 until August, 1991, David was employed by Teradyne, Inc., in Boston, Massachusetts. At Teradyne he learned the C language and developed software relevant to the computer simulation of digital logic hardware. In July, 1987, David married Enid Yvonne Kaufman, a classmate at the Bronx High School of Science (Bryn Mawr College 1983, University of Miami 1992). The couple have two daughters, Jane and Polly.

David finally succumbed to the call of even higher education, and in August 1991 he arrived in Ithaca with his family and entered the Ph.D. program in computer science at Cornell University. David has been employed at Bolt, Beranek, & Newman in Cambridge, Massachusetts since August 1996.
To my wife, Enid, and to my parents.
Acknowledgments

This dissertation would not have been possible without the support —intellectual, financial, and moral— of many people.

Ken Birman’s leadership and support, both as my chairman and as head of the Horus research group, enabled me to continue my work at Cornell during the past four years. His wisdom and insight steered me away from many a wrong path.

Robbert van Renesse provided essential guidance during the application of my methods to useful and interesting properties of the Horus system, and invaluable advice on the seemingly straightforward yet also overwhelming task of organizing this dissertation in its written form. Among many other qualities of a good advisor, Robbert had the patience to read the many written drafts I produced at various times. The results of Robbert’s own work contributed directly to my success as well: I was able to hone some of my thinking by working in the experimental “Horus minus minus” (Horus——) system that he developed, and his development of bypass protocols helped enable the development of Ensemble, a Horus architecture in which layers are coded in the functional language ML, which in turn helps close the gap between my theory and the practice of Horus development. Finally, I am
grateful to Robbert for the idea of depicting Horus protocol layers in the shape of LEGO™ bricks.

Tim Teitelbaum helped me gain a broader perspective on the fundamental problems and approaches to them. Roger Farrell guided me through a minor in mathematics — my major subject in my undergraduate education — allowing me to enjoy once again the beauty of pure mathematics, but at a high level that few are privileged to experience.

I owe Fred Schneider a great debt for many good turns, beginning with his course in formal methods. Fred was a strong advocate of the use of the Temporal Logic of Actions and the creation of a live demonstration. Without these I might still be stumbling in the dark, and certainly would not have achieved nearly as satisfying an outcome. Fred also provided invaluable critical feedback at various stages of my work.

Roy Friedman engaged in many helpful discussions of various aspects of Horus, both the practical and the deeply theoretical.

The work reported in this dissertation was supported in part by ARPA/ONR grant N00014-92-J-1866. Any opinions, findings, or recommendations presented in the following pages, however, are my own and do not necessarily reflect the views either of the Advanced Research Projects Agency or of the Office of Naval Research.

I hesitate to single out fellow students, since I can hardly thank all of them properly in this space. I must, however, thank Mark Hayden for developing the first ML-language version of Horus and for other happy innovations that have
enhanced the value of my own results. I wish also to thank just a few of those whose friendship I’ve enjoyed over the past five years: Guerney D.H. Hunt for many motivational words, Pei-Hsin Ho for putting up with me as an office mate for three years, and Lorenzo Alvisi for being Lorenzo.

I would like to thank Sam Toueg, sometime Graduate Faculty Representative in our department, and Hillary Ford, Assistant Dean in the Graduate School, for helping me survive the first three semesters of my graduate degree work, and Nick Trefethen, Sam’s successor, for helping me persevere through the final year.

I would like to thank the technical and administrative support staff in the Department of Computer Science for keeping the computers running and the paperwork flowing. They helped make smooth sailing out of many tasks that otherwise would have been unthinkably difficult. My thanks go to all the faculty as well for their academic offerings, from which I’ve benefited greatly, and for all the obvious and less obvious efforts by which they created the opportunities for education and research here.

I am grateful to the late Godtfred Kirk Christiansen, President of the LEGO Group in Denmark during the 1960s, for making LEGO™, perhaps my favorite childhood toy, available in the United States, and to Stephen Gustavson for figuring out how to draw LEGO™ pieces beautifully using PostScript. Judging from the literature in computer science, many of us who now build systems from lines of code started out building towers out of little plastic bricks. Who would have thought that decades after I first built a wall of the “automatic binding bricks,” I would be using pictures of them to illustrate my doctoral dissertation?
To my parents, Elsa C. Karr and the late Ronald L. Karr, I owe a profound debt, for bringing me into the world, of course, but also for providing a healthy environment and for all their help and encouragement in setting and pursuing goals throughout my life.

Finally, this dissertation could never have come about if not for the sacrifices made by my wife, Enid, who abandoned the comfort and security of a life near family and friends to move with two young children to an unfamiliar place and endure five years of my graduate studies. I am forever grateful for all the love and support I have received from her and from our daughters Jane and Polly.
Table of Contents

Biographical Sketch iii
Dedication v
Acknowledgments vi
List of Tables xii
List of Figures xiv

1 Introduction 1
  1.1 Horus 2
  1.2 Inheritance of Protocol Properties 4
  1.3 Summary 6

2 The Temporal Logic of Actions 8
  2.1 The Semantics of TLA 9
  2.2 States, Variables, and Values 9
  2.3 Actions and Temporal Formulas 12
  2.4 Specifications 17

3 The Horus Stack 21
  3.1 Events 24
  3.2 Channels 27
  3.3 The Horus Uniform Group Interface 32
  3.4 Additional Assumptions 34
  3.5 Local FIFO transmission 42

4 Horus Layers and Communication Properties 49
  4.1 Safety Properties 50
    4.1.1 Maximum Message Size 50
4.1.2 Delivery of Correct Messages .................. 51
4.1.3 Message Ordering .......................... 53
4.2 Liveness Properties .......................... 54
   4.2.1 Best-Effort Delivery .................. 54
   4.2.2 Eventual Delivery .................. 56
   4.2.3 Stream Sending .................. 58
4.3 Assume-Guarantee Specifications of Layers .................. 58
4.4 Building a Stack ................................ 64
4.5 Group Communication and Fault Tolerance .................. 68
4.6 Conclusion .................................. 71

5 Mechanical Verification of Protocol Stacks 72
   5.1 Initialization of the Verifier .................. 72
   5.2 Selecting a Network .................. 73
   5.3 Adding Protocol Layers .................. 75
   5.4 Selecting Application Properties .................. 77
   5.5 Using Protocol Layers to Discharge Application Properties .................. 79
   5.6 Experimenting with Different Protocol Stacks .................. 81
   5.7 Dependence on Properties of the Network .................. 81
   5.8 The “Document” Mode .................. 83
   5.9 Notes on the Design of the Verification Program .................. 84
   5.10 Conclusion .................................. 86

6 Design and Verification of Protocol Layers 87
   6.1 Argument by Design and Observation .................. 88
   6.2 Proof by Hand ................................ 89
   6.3 Automated Verification of Layers .................. 91
   6.4 Transforming the Specifications of Layers .................. 92

7 Conclusion 98
   7.1 Related Work ................................ 99
   7.2 Results Obtained and Future Work .................. 102

A Useful Facts in the Temporal Logic of Actions 105
   A.1 The Composition Theorem .................. 105
   A.2 Transforming Assume-Guarantee Formulas .................. 108

B Proof of Properties of a Stack 114

Bibliography 123
# List of Tables

1.1 Names of properties of Horus layers ........................................ 5
1.2 Properties required, provided, and inherited by Horus layers .... 5

2.1 Examples of state functions .................................................... 10
2.2 Examples of state predicates .................................................. 11
2.3 Examples of actions .............................................................. 12
2.4 Examples of temporal formulas .............................................. 16
2.5 Classes of expressions in TLA ................................................ 16
2.6 Specifications of components ................................................. 18

A.1 Proof of Fact 5 by case analysis ............................................ 111
# List of Figures

1.1 LEGO™ block model of a Horus stack ........................................... 2
3.1 A generic Horus stack ......................................................................... 22
3.2 Structure of a Horus stack .................................................................. 23
3.3 Data types of Horus events ................................................................ 26
3.4 Data type of a local channel .............................................................. 28
3.5 An implementation of a channel .......................................................... 31
3.6 Horus Uniform Group Interface .......................................................... 33
3.7 Interleaving assumption for a Horus stack ......................................... 35
3.8 Activity log of a Horus stack .............................................................. 37
3.9 A filter function over the activity log .................................................. 38
3.10 Setting auxiliary variables for use in filter functions ......................... 40
3.11 Full set of additional assumptions for a Horus stack ....................... 41
3.12 Local FIFO delivery and liveness ..................................................... 44
3.13 InterHUGI, a buffer layer between protocol layers ....................... 46
3.14 Insertion of InterHUGI layers in a Horus stack ............................... 47
4.1 Assume-guarantee specification of FRAG ........................................... 60
4.2 Assume-guarantee specification of NFNAK ....................................... 62
4.3 Assume-guarantee specification of SIGN .......................................... 65
4.4 Stacking NFNAK over SIGN over FRAG ................................. 66
5.1 Initial display of the interactive stack verifier .................................. 73
5.2 Selecting the Ethernet network layer ................................................. 74
5.3 Adding the SIGN protocol layer ....................................................... 75
5.4 Adding the NAK protocol layer over SIGN ....................................... 76
5.5 “Turning off” MaxSend(1.5K) ............................................................ 77
5.6 Discharging MaxSend(1.5K) by adding a FRAG(1.5K) layer ............... 78
5.7 An inappropriate substitution of an NFNAK layer for NAK ............... 79
5.8 An “upside-down” stack with good communication properties ....... 80
5.9 A stack inadequately supported by CHKSUM over Ethernet ............. 82
5.10  CHKSUM adequately supported by a Secure network  . . . . . .  83
5.11  Self-documentation of the NAK layer  . . . . . . . . . . . . . .  84
5.12  Self-documentation of the FIFO property  . . . . . . . . . . . .  85
Chapter 1

Introduction

Communication between asynchronous processes via networks has been a large and growing part of the world of computing for decades; the modern Internet, which has lately received so much attention in the popular press, has roots in the ARPANET developed in the 1960s. The notion of adding communication properties to a network by layering multiple levels of protocol code over it has for many years been demonstrated by the OSI protocol stacks that support communication protocols such as UDP and TCP/IP.

Nevertheless, software developers often find it surprisingly difficult to create applications in which the execution of one task is to be distributed over several machines. Even a simple client-server query can become complex if the client cannot be sure the server will ever respond, or if the response might exceed some maximum allowable message size. If distributed processing is desired for other purposes, for example load sharing or replication for improved availability, application developers working over standard communication protocols must perform additional work to coordinate processes. Alternatively, this need can be addressed by means of a higher-level communication system. The design and verification of